SKROLLI 2015.2

SKROLLI

2015.2
SKROLLI.FI

Tietokonekulttuurin erikoislehti

3 **Pääkirjoitus**

4 **Kuvamerkkien uusi aika**
Kuvilla juttelu ja emojit hämärtävät kirjoituksen määritelmää.

9 **Kolumni – Janne Sirén**
Autoilu uusiksi Teslan johdolla.

10 **UNSCII uudistaa**
Miljoonien symbolien merkkigrafiikkaa.

12 **Ei näin!**
Neo Geo -perheen tyyris tarina.

16 **Upeat ja unohdetut Unix-koneet**
Ammattikäytöstä hyljätyt koneet herättävät harrastajissa intohimoja.

20 **Speedrunien vimma**
Näihin suorituksiin ei ihminen pysty.

22 **Mittausdataa Jollasta**
Sailfish-ohjelmointisarjan toisessa osassa antureiden ja kompassin käyttö.

25 **Croutonilla lisäkiiltoa Chromebookiin**
Kepeästä kattavaan käyttökseen ja takaisin.

28 **Verkkovakoilun tekninen ulottuvuus**
Isoveli kyttää kaikkialla – näin rihmatkin viedään yltäsi.

36 **Konekielen kiemurat**
Jos sen haluaa tuntea, sen kanssa on pakko jutella.

44 **Aikavälikertaus muistin tukena**
Jäikö vitsa vääntämättä? Aikavälikertaus auttaa!

48 **Unity 3D**
Pirteistä pikkupeleistä komeaan kolmedeehen – Unity taipuu moneen.

52 **Raspista vanhan ajan kotitietokone**
RISC OS Picon avulla halpa nykyrauta taipuu perinteiseen Basic-ohjelmointiin.

57 **Varmuuden hinta**
Sertifikaateilla varmistetaan luottamuksen ketju.

60 **Ohjelmoi Oculus Riftille**
Unelmien virtuaalimaailma omaan silmikkoon? Onnistuu!

64 **Mikrokivikausi 30 vuotta**
Testikuvia erään sarjan matkalta.

66 **Satunnainen samooja**
Nordic palaa luolastoihin.

70 **Pelimuseo syntyy hartiavoimin**
Joukkorahoitetaan Tampereesta Suomen pelipääkaupunki!

71 **Skrollitöllö**
Juttelimme peleistä ja tekniikasta teekkareiden televisiostudiossa.

Ville-Matias Heikkilä
päätoimittaja

Auta nettiriippuvaista kansaa!

Useimmilla lukijoilla lienee kokemusta peleistä tai nettipalveluista, joista ei meinaa päästä irti millään. Moni on varmasti myös kamppaillut vakavamman addiktion kanssa.

Vuosituhannen vaihteen aikoihin kiinteiden laajakaistayhteyksien saatavuus parani huomattavasti. Oma suhteeni nettiin muuttui silloin ongelmalliseksi.

Koska yhteysminuutteja ei enää tarvinnut laskea, virtuaaliseen ihmemaahan saattoi olla uppoutuneena vaikka koko valveillaoloaikansa. Opiskelu kärsi. Pahimmillaan kaupassa käyminenkin vaati ponnisteluja, kun turruttavasta irc-virrasta piti irrottautua muutamaksi minuutiksi. Vasta vuosien kuluttua sain nettisuhteeni kunnolla ruotuun kokeiltuani useita erilaisia itsekurituskeinoja. Rajoitan edelleen yhteys- ja tavoitettavuusaikojani, vaikkei pakottavaa irkinkyttäystarvetta olekaan enää vuosiin ollut.

Sittemmin koko kansa löysi ympärivuorokautisen nettisosiaalisuuden ilot. Nettiriippuvuuden kanssa nuoruudessaan kamppailleelle tämän seuraaminen oli vaivaannuttavaa. Tuttava toisensa jälkeen huumaantui Facebookista, koukuttui siihen ja alkoi osoittaa samoja riippuvuuden merkkejä kuin irc-addiktit vuosia aiemmin. Samalla riippuvuudesta on tullut yhä vaikeammin havaittavaa. Netti kulkee taskussa mukana, ja runsaasta sosiaalisen median käytöstä on tullut suorastaan normi, josta poikkeavaa oudoksutaan. Eikä asiaa auta se, että suositut nettipalvelut on alun alkaenkin suunniteltu koukuttamaan käyttäjänsä – eiväthän ne muuten olisi suosittuja.

Koukuttavia viestimiä on toki ollut aiemminkin. Television turruttavasta ja passivoivasta vaikutuksesta on paasattu jo vuosikymmeniä. Netissä roikkuminen on vuorovaikutteisen luonteensa vuoksi pienempi paha kuin telkkarin edessä löhöäminen, mutta sen ongelmia ei osata vielä tunnistaa samalla tavoin. Tästä syystä kansa tarvitsee kaikkien niiden apua, jotka ovat kohdanneet nettiriippuvuutta jo kauan sitten.

Jos olet kärsinyt nettiriippuvuudesta, niin avaudu kokemuksistasi! Jos huomaat kavereissasi tuttua oirehdintaa, niin ota asia puheeksi! Kansan on jo korkea aika päästä kollektiivisesta somehuumastaan vaiheeseen, jossa se tunnistaa ongelmansa ja hakee siihen apua.

Skrolli

Tietokonekulttuurin erikoislehti

Yhteydenotot toimitus@skrolli.fi
Ircnet: #skrolli

Päätoimittaja Ville-Matias Heikkilä
Toimituspäälliköt Toni Kuokkanen & Ninnu Koskenalho
Taiteellinen johtaja Risto Mäki-Petäys
Kuvatoimittaja Mitol Meerna
Mediamyynti Jari Jaanto
Talous Anssi Kolehmainen

Muu toimitus Mikko Heinonen, Jukka O. Kauppinen, Ronja Koistinen, Sade Kondelin, Teemu Likonen, Annika Piiroinen, Manu Pärssinen, Antti Pöllänen, Suvi Sivulainen, Kalle Viiri

Tämän numeron avustajat Tapio Berschewsky, Yrjö Fager, Eric Hartin, Tero Heikkinen, Chris Helenius, Nordic the Incurable, Jukka K. Korpela, Miikka Lehtonen, Sakari Leppä, Asser Lähdemäki, Jussi Määttä, Markus Ottela, Manu Pärssinen, Mikko Rasa, Markku Reunanen, Minna Sarakontu, Janne Sirén, Outi Sorsa, Mikko Torvinen, Jari Viitala, Wallu

Julkaisija Skrolli ry

Painopaikka Tammerprint, Tampere,
ISSN 2323-8992 (painettu)
ISSN 2323-900X (verkkojulkaisu)

Etukannen kuva:
Wallu

YMPÄRISTÖMERKKI
MILJÖMÄRKT
441 878
Painotuote

Kuvamerkkien uusi aika

Emojit ja muut kuvamerkit ovat tulossa mukaan viestintäämme, ja kirjoitettu kieli saa kuvakirjoituksen piirteitä. Tämä muuttaa jopa kirjoituksen käsitettä: sanan ja kuvan ero hämärtyy.

Teksti: Jukka K. Korpela

Kuvat: Mitol Meerna, Jukka K. Korpela, Minna Sarakontu

Kirjoitetussa viestinnässä on aina käytetty pieniä, noin kirjaimen kokoisia kuvia tekstin joukossa. Tosin menettelyn yleisyys on vaihdellut, ja sitä vähensivät sellaiset viestintämuodot, joissa ei voinut käyttää kuvia. Tekniikka on nyt taas mahdollistanut kuvamerkkien helpon käytön. Se on jo levinnyt laajalle monissa nykyajan viestintämuodoissa, kuten kuva 1 havainnollistaa.

Kuvamerkkien käyttö on osittain jo varsin vanha ilmiö. Siinä, mitä nyt on tapahtumassa, on kuitenkin kaksi olennaista uutuutta. Ensinnäkin kuvamerkkejä käyttävät yhä useammat ihmiset yhä useammissa yhteyksissä, ja merkkien monipuolisuus lisääntyy. Toiseksi kuvamerkkejä on ruvettu laajamittaisesti standardoimaan ja määrittelemään nimenomaan merkeiksi, joilla on omat koodinsa merkkikoodissa (Unicodessa) ja joita siksi voidaan käyttää tekstissä kuten kirjaimia.

Ikonista symboliksi

Filosofi Charles S. Peircen esittämässä analyysissa erotetaan toisistaan merkkien (signs) eri lajeja seuraavasti:

- *ikoni* on kohteensa kuva eli näkyvästi sen kaltainen; esimerkiksi kuva nuotiosta esittää tulta
- *indeksi* viittaa kohteeseen syy-yhteyden kautta; esimerkiksi savu viittaa tuleen
- *symboli* viittaa kohteeseen vain, koska se on otettu sellaiseen käyttöön ja se on vakiintunut tai siitä on sovittu; esimerkiksi suomen sana ”tuli” tai englannin sana ”fire” on tässä mielessä symboli.

Ikonille on ominaista, että sen voi ymmärtää intuitiivisesti: se esittää kohdettaan graafisesti. Indeksin ymmärtäminen sen sijaan riippuu vahvasti kokemuksesta ja kulttuuristakin. Symboli on täysin sovinnainen, joskin sovinnaisuuden aste vaihtelee. Esimerkiksi kiinalaisen kirjoituksen elementti ”⺦” (CJK RADICAL FIRE) on sinänsä sovinnainen, mutta siinä voi kyllä nähdä tulen ikonin tai indeksin (hiilloksen).

Sanaa ”symboli” voidaan käyttää myös hyvin laajassa merkityksessä niin, että myös ikonit, indeksit, tavalliset kuvat, esineet, eleet, sanat ym. voivat olla symboleita. Osasyy tähän on, että suomen kielessä ei ole Peircen käsitteistön sanaa ”sign” vastaavaa termiä vaan sana ”merkki” tarkoittaa tavallisimmin kirjoitusmerkkiä (character).

Ikonin ja indeksin ero on suhteellinen. Ne voivat myös kehittyä symboleiksi, kuten on käynyt esimerkiksi levykettä esittävälle ikonille. Sitä käytetään tietokoneohjelmissa yleisesti

tarkoittamaan Tallenna-toimintoa, vaikka levykkeiden käyttö on loppunut. Monet eivät ole koskaan nähneet levykettä saati käyttäneet sitä, mutta tunnistavat silti levykesymbolin tarkoittavan tallentamista (kuva 2).

Hymiöiden tulo

Hymiöt (smileys, emoticons) otettiin käyttöön 1980-luvulla viestinnän elävöittämiseen tilanteessa, jossa voitiin käyttää vain merkkipohjaista viestintää. Hymiöiden viehätystä lisäsi se, että niiden ymmärtäminen ja käyttö vaatii oivaltamista tai oppimista. Merkkiyhdistelmä ”:-)” näyttää hymyileviltä kasvoilta vain, jos sitä katsoo pää vasemmalle kallistettuna. Teoreettisessa luokituksessa hymiötä voidaan kuitenkin pitää ikonina.

Kuva 1. Kuvamerkkien avulla juttelua Whatsappissa.

Netistä löytyy lukemattomia hymiöiden kokoelmia, joissa useimmat ovat pitkälti tuntemattomia ja täysin tarpeettomia. Usein kyse on vain graafisesta ilottelusta: kehitellään merkkiyhdistelmiä, jotka voisivat tarkoittaa jotain (kuva 3).

Hymiöt ovat erikoistapaus *merkkigrafiikasta*, jota on käytetty ja käytetään keinovalikoiman todellisten tai luuloteltujen rajoitusten takia. Merkkigrafiikassa käytetään merkkiyhdistelmiä, jotka voidaan hahmottaa kuvioiksi. Jopa merkin ”>” tai merkkiparin ”->” käyttöä voidaan pitää merkkigrafiikkana: niitä käytetään niiden ulkoasun takia oikeamman symbolin kuten nuolen ”→” sijasta. Merkkigrafiikasta käytetään myös nimitystä Ascii-grafiikka tai Ascii-taide, silloin kun merkeistä sommitellaan isoja kuvioita.

Hymiöistä kuvamerkkeihin

Jo jonkin aikaa monet ohjelmat ovat muuntaneet hymiöitä kuviksi esimerkiksi niin, että kun käyttäjä kirjoittaa kolme merkkiä ”:-)”, ohjelma sisäisesti muuntaa ne kuvaksi tai kuvamerkiksi tai vain näyttää sen tilalla hymynaaman.

Alkuperäisen idean mukaiset hymiöt ovatkin hitaasti väistymässä, ja perussyykin on ilmeinen. Kuvat ja kuvamerkit ovat paljon luonnollisemman näköisiä kuin hymiöt, ja niihin saa paljon enemmän vaihtelua: 😄 😉 😍…

Kaikkein tavallisimmat hymiöt saattavat kuitenkin jäädä käyttöön, koska ne ovat vakiintuneet kirjoituksen osaksi. On jopa kysytty kielitoimistolta, miten hymiöiden käyttö vaikuttaa välimerkkien käyttöön – siis kysytty muodollisia sääntöjä aika epämuodollisten ilmausten käyttöön. Kysymykseen myös vastattiin: ”Selvyyden vuoksi hymiö kannattaa sijoittaa virk-

Kuva 2. Levykkeen kuva oli alkujaan ikoni, mutta nykyisin se on yleensä tallentamisen symboli.

O:-) Angel
0;-) Angel wink - male
:-{{ Angry Very
~:o Baby
d:-) Baseball
:-{0 Basic Mustache

Kuva 3. Ote Netlingo-sivuston hymiösanastosta.

Kuva 4. Muutamia Unicoden version 7 mukanaan tuomista 250 uudesta emojimerkistä.

Kuvamerkki – siis mikä?

Kuvamerkki on kirjoitusmerkki, joka esittää jotain asiaa tyylitellyn kuvan tavoin. Sitä voidaan käyttää tekstissä kuten muitakin merkkejä, joiden merkitys on olennaisesti graafinen, kuten ”⌛” (tiimalasin merkki, HOURGLASS U+231B).

”Kuvamerkki” ei ole vakiintunut termi, mutta hyödyllinen. Se ei myöskään ole täsmällinen käsite, koska merkin kuvallisuus on tulkinnanvaraista. Kuvamerkeiksi voidaan tulkita ainakin

- wingdings-merkit, esimerkiksi ✂✈★, jotka on nykyisin koodattu myös Unicode-merkeiksi
- emoji-merkit kuten 🍲🚾
- sekalainen valikoima muita merkkejä, esimerkiksi ⌂ (house).

Jos tiedostossa olevassa tekstissä on kuvamerkki, se tallentuu muiden merkkien tavoin omalla koodillaan. Esimerkiksi ”☺” on tavallinen merkki Unicodessa: U+263A WHITE SMILING FACE. Kuvamerkkiin vaikuttavat fonttiasetukset kuten muihinkin merkkeihin: se voi muuttaa kokoa, väriä ja fonttia.

Kuva 5. Kulttuurisidonnaisia emoji-merkkejä. Samat merkit kahdella fontilla: CLOSED MAILBOX WITH LOWERED FLAG, MONEY WITH WINGS ja FATHER CHRISTMAS.

Kuva 6. Sateenkaarta esittävä Unicode-merkki värillisenä gemojina ja mustavalkoisena, vaikeammin hahmottuvana glyyfinä.

Kuva 7. Tulevaisuudessa voimme ehkä valita kuvamerkkien "ihonvärin".

keen päättävän välimerkin perään" (Kielikello 4/2009).☝

Emoji-merkit

Sana "emoji" liitetään usein sanaan "emootio", mutta yhteys on satunnainen. Sana "emoji" tarkoittaa japanissa kuvamerkkiä. Emojit tulivat laajaan käyttöön ensin japanilaisissa tekstiviesteissä.

Jotkin emojit ovat hyvin kulttuurisidonnaisia, Japanin kulttuurista riippuvia, mutta suuri osa on yleisinhimillisiä. Emoji-merkin käsite ei ole täsmällinen, vaan on eri tulkintoja siitä, mitä kuvamerkkejä pidetään emoji-merkkeinä.

Emojien yhdestä osajoukosta käytetään Unicode-standardissa nimeä "emoticons", mutta ne esittävätkin kasvojen ilmeitä ja eleitä, esimerkiksi 😿 (WEARY CAT FACE).

Emojit ovat luonteeltaan yleensä vahvasti ikonisia: ne esittävät jotakin näkyvää asiaa. Toisaalta samaa kohdetta esittäviä emoji-kuvia on usein paljon erilaisia, jolloin kuvan valinnalla ilmaistaan myös asenteita ja tunteita. Viestit voivat täten olla hyvin tiiviitä, mutta tulkinnanvaraisia. Tulkinnanvaraisuutta lisää se, että emojit on usein tarkoitettu indekseiksi. Esimerkiksi emoji, jossa on lautanen, veitsi ja haarukka, tarkoittaa yleensä pikemminkin ruokailua kuin näitä kolmea esinettä.

Kuvamerkkien kulttuuririippuvuus

Merkin ja kuvan keskeinen ero on, että merkin ulkoasu riippuu fontista, joskus paljonkin. Vaikka esimerkiksi tiimalasin merkki tarkoittaa vain tiimalasia, ei esimerkiksi jotain abstraktia ajan kulumisen ideaa, sen ulkoasu voi vaihdella. Todellisia tiimalaseja on erilaisia, samoin tyyliteltyjä tiimalaseja. Sen sijaan kuva on kiinteä.

Kuvamerkkien ulkoasu voi kuitenkin vaihdella suuresti (kuva 5). Periaatteessa kuvamerkki ei ilmaise abstraktia ideaa vaan jotakin konkreettista ja näkyvää. Käytännössä kuvamerkin muoto voi vaihdella fontin mukaan paljonkin. Lisäksi kuvamerkki voi olla olennaisesti riippuvainen kulttuurista. Esimerkiksi erilaiset postilaatikkomerkit esittävät amerikkalaisia laatikoita. Niissä on olennaista, onko laatikkoon kiinnitetty lippu alhaalla vai ylhäällä, mutta tämä ei välttämättä sano mitään ihmiselle, joka ei tunne amerikkalaista käytäntöä.

Monitulkintaisuuden ongelma

Vanhassa kuvakirjoituksessa käytettiin eri tekniikoita kuvien rajoitusten ja monitulkintaisuuden hallitsemiseksi. Osaa niistä käytetään yhä. Kuvamerkkiin voidaan liittää sanoja tai lyhenteitä selventämään niitä. Vanhempi tapa on liittää kuvamerkkiin muita graafisia merkkejä, jotka jollain sovinnaisella tavalla rajoittivat merkitystä. Vielä yksinkertaisempaa on käyttää useita kuvamerkkejä yhden käsitteen ilmaisemiseen.

Kuvakirjoituksen uudessa tulemisessa tekniikat pysynevät yksinkertaisempina. Koska kirjainkirjoitus on jo keksitty, on luonnollista käyttää selittäviä sanoja. Vielä yksinkertaisempaa on, että kuvilla esitetään vain sellaisia asioita, että ne ovat asiayhteydessä yksikäsitteisiä; muu esitetään sanoilla. Lause "Nyt mä 🚙🍏 ja sit 🍔🍟" toimii hyvin, koska lyhyet pikkusanat tekevät selväksi, että kuvataan omia aikomuksia eikä esimerkiksi tehdä ehdotusta; tämän tekeminen pelkillä kuvamerkeillä olisi varsin vaikeaa, vaikka kuvamerkit ilmaisevatkin toiminnan sisällön hyvin.

Kuvamerkkejä standardoidaan

Merkkikoodien määrittelyssä, joka on nykyisin olennaisesti Unicode-standardin kehittämistä, on aiemmin suhtauduttu nihkeästi kuvamerkkeihin. On koodattu ensisijaisesti vain sellaisia symboleita, joilla on todistettavissa olevaa käyttöä merkkeinä tekstissä, ei vain graafisina alkioina esimerkiksi julisteissa. On ajateltu, että jos mitä tahansa graafisia symboleja ruvetaan lisäämään merkkikoodiin, päädytään suureen sekaannukseen ja lopulta koodiavaruuden loppumiseen. Unicode-standardissa on noin miljoona koodipaikkaa merkeille, ja vakaaksi tarkoitettu päätös on, että koodiavaruutta ei tästä laajenneta.

Nyt kuitenkin Unicodeen on lisätty suuri määrä kuvamerkkejä, ja lisää on tulossa. Unicoden versiossa 7 (kesällä 2014) tuli 250 uutta kuvamerkkiä (emoji-merkkiä), joista on muutama näyte kuvassa 4. Kuvamerkkien virta Unicodeen on siis avattu. Tosin kovin nopeaa merkkien lisääminen ei voi olla. Unicode-standardin version 7 julkistuksen yhteydessä päätettiin, että uusi versio julkaistaan vuosittain kesäkuussa. Lisäksi voidaan julkaista väliversioita. Vuonna 2016 julkaistavaan versioon 9 on ehdolla vain 38 uutta kuvamerkkiä.

Unicoden versiossa 7 merkkeihin tulivat värit. Unicode nimittäin määrittelee nyt kuvamerkeille mahdollisuuden sekä mustavalkoiseen esitykseen (tai oikeastaan yksiväriseen) että moniväriseen esitykseen (gemoji), kuten kuva 6 havainnollistaa.

Unicode määrittelee myös tavan valita tällaisten esitystapojen välillä. Keinona on teknisesti se, että emoji-merkkiä seuraava Unicode-merkki VARIATION SELECTOR-16 (U+FE0F) on pyyntö esittää emoji-merkki ”emoji-tyyliin”, merkki VARIATION SELECTOR-15 (U+FE0E) taas pyytää ”tekstityyliä”. Tosin toistaiseksi melko harvat ohjelmat tukevat näitä tekniikoita. Tavallisempaa on, että ohjelma käyttää kiinteästi jompaakumpaa tyyliä.

Tasa-arvoa merkkeihin

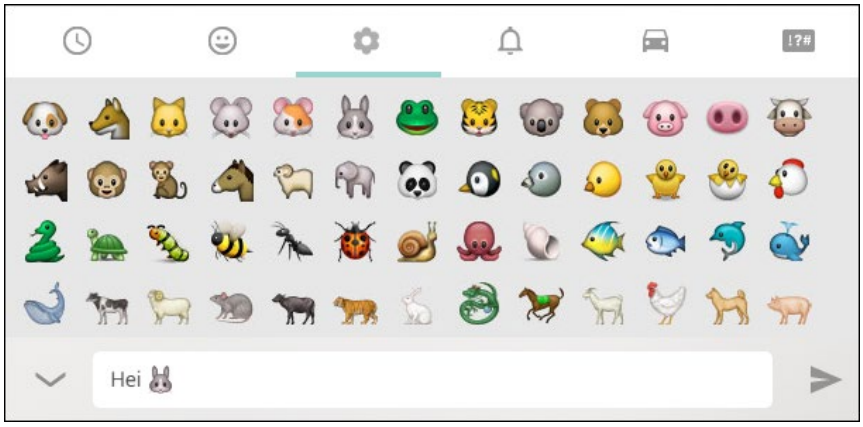
Edellä mainittiin, että Emoji-merkeillä voi olla myös moniväriset ulkoasu yksivärisen ohella. Tämä on lisännyt sitä ongelmaa, että ihmistä esittävässä merkissä on määrätty ihonväri, joka on käytännössä lähes aina ollut vaalea. Tämän takia on laadittu luonnos, joka määrittelee kuvamerkkeihin yhdistettäviä väritysmerkkejä, ”skin tone modifiers” (kuva 7).

Väritysmerkit eivät tarkoita yksinkertaista värittämistä, vaan ne voivat muuttaa sekä ihon että tukan väriä. Niille on annettu abstrakteja nimiä kuten EMOJI MODIFIER FITZPATRICK TYPE-5.

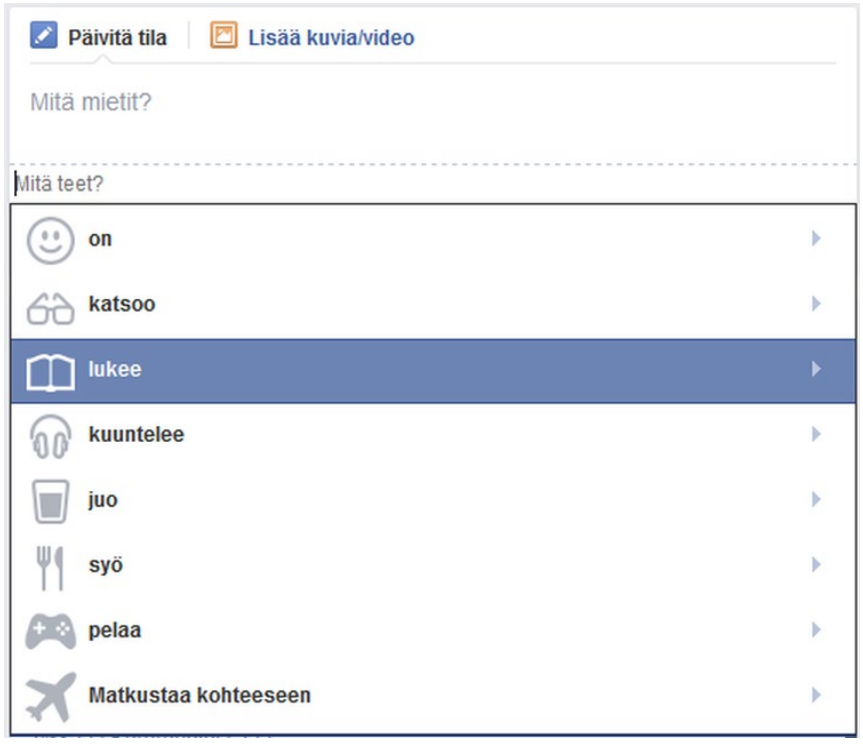
Kirjoittamisen tekniikoiden paljous

Kuvamerkkejä voi kirjoittaa kaikilla merkkien kirjoittamisen yleisillä tekniikoilla, kuten järjestelmän merkkipaletista valitsemalla tai merkin Unicode-numeroa käyttävillä tekniikoilla. Eri ohjelmissa voi olla omia merkkipalettejaan, jotka voivat olla ryhmiteltyjä tai hierarkkisia (kuva 8). Lisäksi joissakin ohjelmissa emojeja voi kirjoittaa Unicode-nimiensä avulla esimerkiksi siten, että :rocket: tuottaa merkin ”🚀”, jonka Unicode-nimi on ROCKET. Kuvamerkkejä voi myös kopioida ja liittää tekstistä toiseen, esimerkiksi verkkosivulta getemoji.com sähköpostiin tai Facebook-päivitykseen.

Ohjelmat saattavat tarjota hyvinkin aktiivisesti mahdollisuutta kuvalliseen ilmaisuun tekstin yhteydessä. Esimerkiksi Facebookissa voi tilapäivityksen helposti aloittaa kuvakkeella (kuva 9). Tällöin muut voivat heti nähdä, minkätyyppisestä päivityksestä on kyse. Kuvien tunnistaminen on nopeampaa kuin tekstin lukeminen. Lisäksi kuva vetää huomion puoleensa.



Kuva 8. Whatsappin hierarkkinen kuvamerkkivalikko: yläriviltä valitaan aihepiiri, esimerkkitapauksessa eläimet, kasvit yms.



Kuva 9. Facebookin tilapäivityksessä saa hymiökuvaketta napsauttamalla listan, josta voi valita kuvan viestinsä alkuun.

Fontit rajoittavat

Uutta merkkiä ei voi esittää, ennen kuin se on toteutettu johonkin fonttiin. Voi kestää hyvin kauan – kymmenen vuottakin –, ennen kuin Unicodeen lisätty merkki on useimpien käyttäjien saatavilla. Tämä on tärkein syy, joka rajoittaa kuvamerkkien käyttöä digitaalisessa viestinnässä ja johtaa usein kuvien käyttöön kuvamerkkien sijasta.

Tietokoneissa olevat fontit eivät yleensä päivity yhtä helposti kuin käyttöjärjestelmät ja selaimet. Jokaisen käyttäjän on pitänyt erikseen hankkia eri fonttien uudet versiot ja uudet fontit. Käytännössä fonttien valikoima riippuu ensisijaisesti käyttöjärjestelmän versiosta, toissijaisesti siitä, mitä sovellusohjelmia on asennettu, koska fontteja tulee myös niiden mukana.

Tilanne on kuitenkin muuttumassa. Microsoft on toteuttanut muun muassa Segoe UI Symbol -fontin laajennetun version automaattisen päivittämisen järjestelmäpäivitysten osana. Kyseinen fontti sisältää suurehkon valikoiman kuvamerkkejä.

Verkkosivuilla on mahdollista käyttää web-fontteja eli sivun mukana automaattisesti latautuvia fontteja. Sama koskee HTML5-sovelluksia eli Javascriptillä tehtyjä, HTML:ää ja CSS:ää käyttäviä, web-selaimessa suoritettavia sovelluksia. Tekijänoikeus-

syistä kyseeseen tulevat yleensä vain vapaat fontit. Uusia emoji-merkkejä on toistaiseksi vain parissa vapaassa fontissa (Quivira, Symbola).

Kirjoitus palaa juurilleen

Kirjoituksen alkumuoto oli kuvakirjoitus. Varhaisimmat säilyneet kirjoitukset ovat paljolti varastokirjanpitoa ja veroluetteloita, joissa esitetään konkreettisia esineitä. Kuvat riittivät siihen, mutta käytännön tarve pakotti yksinkertaiseen, viitteelliseen esitykseen. Ei piirretty taloa vaan parilla vedolla taloa muistuttava kuvio.

Kirjainkirjoitus syntyi, kun muutamia merkkejä ruvettiin käyttämään äänteitä tarkoittamassa. Esimerkiksi talokuvio, joka oli totuttu lukemaan esimerkiksi ”bet” (seemiläinen taloa tarkoittava sana), otettiin b-äänteen merkiksi. Ehkä syynä oli aluksi tarve kirjoittaa ihmisten ja jumalien nimiä, koska niitä oli vaikeaa tai mahdotonta esittää kuvamerkeillä. Kirjanpidon kaltaisten tekstien ohella vanhimpia tekstejä ovat lyhyet rukoukset ja muut uskonnolliset tekstit sekä omistamista osoittavat merkinnät.

Kirjainkirjoitus tarjosi myös mahdollisuuden ilmaista paljon monimutkaisempia ja käsitteellisempiä asioita kuin kuvakirjoitus. Sillä voitiin sanoa kaikki, mikä voidaan sanoa puhutulla kielellä. Tämä päti siitä huolimatta, että varhaisin kirjainkirjoitus oli monin tavoin puutteellista. Siinä merkittiin vain konsonantit ja nekin epäjohdonmukaisesti: sama kirjain saattoi olla kahden eri konsonantin merkki. Kirjoituksen ei tarvitse olla täydellistä ollakseen hyödyllistä.

Kuvakirjoitus ei ole koskaan kadonnut, vaan se on säilynyt kirjainkirjoituksen ohella, esimerkiksi opasteissa joko yksinään tai tekstin lisäksi. Erikoisaloilla saattaa juokseva tekstikin yhä sisältää kuvakirjoitusainesta kuten halkaisijan merkki ”⌀” ja suuntaa osoittava nuoli ”→”. Lisäksi on symboleja, jotka eivät ole graafisesti kuvailevia vaan sovinnaisia mutta luonteeltaan kielestä riippumattomia ja enemmänkin käsitteen kuin sanan merkkejä, kuten ”€” ja ”%”.

Miksi kuvakirjoitus on nyt yleistymässä? Kuvakirjoituksen yksi ilmeinen etu on kieliriippumattomuus, mutta emoji-merkit ovat erityisen suosittua japaninkielisessä viestinnässä, jossa kielimuuria ei ole. Miksi japanilaiset ovat ottaneet neljän kirjoitusjärjestelmänsä (kanji, hiragana, katakana ja latinalainen) rinnalle vielä aivan uuden, ikään kuin tuhansissa kanji-merkeissä ei olisi tarpeeksi opettelemista? Miksi emojit ovat levinneet muuallekin ja ovat yleisiä yhteyksissä, joissa voisi hyvin käyttää yhteistä kieltä?

Kuvakirjoituksen yksi etu on, että se ei vaadi luku- eikä kirjoitustaitoa – eikä nykyisin edes piirustustaitoa, koska merkkejä voidaan poimia valikoista. Tämäkään ei riitä selitykseksi, koska kuvakirjoitusta käyttävät ihmiset tyypillisesti osaavat ihan hyvin lukea ja kirjoittaa ja ovat usein hyvin koulutettuja.

Visuaalinen havainnollisuus tuntuu muodostuneen keskeiseksi tavoitteeksi, varsinkin tunteiden ilmaisussa. Olemme tottuneet lukemaan kuvitettuja tekstejä ja pidämme niistä. Lyhyitä viestejä kirjoitettaessa aitoon kuvittamiseen ei yleensä viitsitä ryhtyä, vaikka se olisikin monin tavoin helpompaa kuin ennen. Sen sijaan ikoneilla voi säestää viestiä tai jopa esittää koko viestin.

Viestinnän murros vai ohimenevä muoti?

Kuvamerkit ovat usein hauskoja ja piristäviä, ja monet ohjelmat suorastaan tyrkyttävät niitä kirjoittajille. Olisi helppoa ajatella, että kyseessä on muoti-ilmiö, johon väsytään melko pian. Toisaalta se on jatkunut jo kauan monissa yhteyksissä ja on niistä leviämässä laajemmalle.

Kuvamerkit ovat näennäisen helppoja, mutta kuvakirjoitus ei ole ollenkaan niin yksiselitteistä ja helposti luettavaa kuin voisi luulla. Usein ne ilmaisevat yksinkertaisia konkreettisia esineitä tai asioita taikka esittävät tyylitellysti tunteenilmauksia. Viesti ehkä menee jotenkin perille, mutta ei ole mitään takeita siitä, että vaikutus on haluttu. Kuvia tulkitaan eri tavoilla, kuten sanojakin.

Kuvakirjoitus ei välttämättä suju kovinkaan nopeasti verrattuna sanojen kirjoittamiseen. Tämä tosin riippuu suuresti siitä, esittääkö ohjelma käyttäjälle kontekstista riippuvia valikkoja, joissa on sellaisia kuvamerkkejä, joita hän todennäköisesti haluaa käyttää.

Kuvamerkit eivät ole hetken huumaa, mutta eivät ne itsessään myöskään merkitse vallankumousta. Sen sijaan niillä on oma osansa viestinnän suuressa murroksessa, jossa tavallinen viestintämme muuttuu moni-ilmeisemmäksi. Teksti, tekstin muotoilut, kuva, ääni ja liikkuva kuva muodostavat keinovalikoiman, jota käytetään aina tilanteen mukaan. Alamme jo tottua siihen, että arkipäiväiseen viestiinkin voi liittää saman laitteen kameralla otetun kuvan. Tulevaisuudessa kynnys myös tekstin typografiseen muotoiluun, kuvien piirtämiseen, äänitteiden ja videoleikkeiden lisäämiseen alenee tai häviää.

Kuvamerkeillä on osansa tässä kokonaisuudessa. Kun puolisolle lähetetään ostoslista, voidaan mukaan laittaa vaikkapa kuva halutusta tuotteesta – tai yksinkertaisimmassa tapauksessa vain 🍌🍌🍌🍍🍏🍏. 🏠

Haaste esteettömyydelle

Kuvien käyttö on aina ongelmallista sokeille ja usein myös heikkonäköisille. Koska kuvamerkit ovat yleensä pienikokoisia, niiden tunnistaminen on vaikeaa, jos näkö ei ole terävä.

Näkövammaisten avuksi on kehitetty apuvälineitä kuten puhesynteesi ja Braille-laitteet, jotka esittävät tekstin pistekirjoituksella. Kuvamerkkien Braille-esitystä ei kuitenkaan ole määritelty. Puhesynteesissäkin ne tuottavat ongelmia. Ilmeisin ratkaisu olisi esittää kuvamerkin tilalla sen nimi, mutta tätäkään ei vielä ole toteutettu.

Keksi pyörä uudelleen

Sähköauto antoi odottaa itseään pitkään. Sitten uusi tulokas Tesla Motors uskalsi kyseenalaistaa nykytiedon.

Janne Sirén

”It isn’t what you don’t know that gets you into trouble. It is what you know, but just ain’t so.” Osuva oivallus pätee myös teknologian murroksiin, kuten sähköauton kehittämiseen. Sekin alkoi sujua vasta, kun ymmärrettiin, että vanhat faktat eivät enää pitäneet paikkaansa.

Vasara ja nauloja

Autojätit ylpeilevät modulaarisilla alustoillaan, joista syntyy auto useaan eri kokoluokkaan. Alustoja yhdistää vuosisatainen, kovassa käytössä ollut konsepti: polttomoottori, vaihteisto, pitkittäinen keskitunneli pakoputkelle tai vetoakselille (ajalta, jolloin vetäviä ja ohjaavia pyöriä ei vielä osattu yhdistää) sekä bensatankki.

Kun miljardi-investointeja polttomoottorialustoihin tehnyt autotehdas tarttuu sähköautoon, tulos kuulostaa Saksasta hiljattain tihkuneelta suunnitelmalta: lukuisia pieniä akkuja ripoteltuna vapaisiin koloihin, keskitunneliin, moottoritilaan ja jopa ovitaskuihin. Sähköauto onkin yleensä pohjimmiltaan vain bensa-auto, jossa polttomoottorijärjestelmän tiloja on uusiokäytetty.

Suuri osa autotehtaiden satavuotisista kulttuureista liittyy pienten räjähdysten millintarkkaan hallintaan teräsmöhkäleen sisällä. Rahavirrat taas muodostuvat polttomoottorien jatkuvasta huollon tarpeesta. Sille, jolla on jo kädessään vasara, kaikki näyttää naulalta. Niinpä sähköautoista tulee bensa-autojen puolivillaisia kopioita, lyhyen matkan golfkärryjä, joissa vähät akut kutistavat tavaratilan.

Piilaaksolaisella Tesla Motorsilla ei ole takanaan sataa vuotta, vasta kymmenkunta. Yritys valmistaa vain sähköautoja ja kokoaa akkupatteristoja. Tesla yhdistelee pieniä litiumioniakkuja suuriksi kokonaisuuksiksi. Ensimmäisessä Tesla Roadsterissa vuonna 2008 akut oli vielä pakattu Lotukselta lainatun korin nelisylinterimoottorin paikalle, mutta siihen yhtäläisyydet loppuvatkin.

Monet polttomoottoriauton totuudet eivät koske sähköautoa, mutta voivat kyllä pilata sen. Tesla Model S on ensimmäinen moderni sähköauto, jonka koriin ei ole suunniteltu polttomoottoria öljysäiliöineen, jäähdyttimineen, vaihteistoineen, putkistoineen, vaimentimineen, katalysaattoreineen ja bensatankkeineen. Klassisen auton osista jatkoon ovat päässeet ainoastaan neljä pyörää, jousitus, ratti ja jarrut.

Autokoulu uusiksi

Model S:n lattian muodostaa sileä akkulevy 300–500 ajokilometrille. Renkaiden välissä on yksi tai kaksi pientä ja yksinkertaista, urheiluauton ripeää sähkömoottoria. Perustuksilleen riisuttu Tesla muistuttaa isoa rullalautaa, jossa on ratti. Rullalauta on kuin tyhjä taulu, jolle rakentaa. Minimaalinen rakenne mahdollistaa myös akuston vaihtamisen autopesulaan mahtuvalla robotilla. Model S:ää onkin kutsuttu automaailman Iphoneksi, sillä se on saanut kilpailevat ratkaisut näyttämään absurdin monimutkaisilta.

Vaikka Model S on vain sulavalinjainen viistoperä, taka- ja etukonteissa on tavaratilaa kuin maasturissa – jopa 1790 litraa. Keulassa on voitu panostaa kolariturvallisuuteen, kun polttomoottorin tilavaatimuksia ei ole. Etupenkkien välissä ei ole vaihteistoa vaan lattia tai lokero, ja takapenkin lattia on kauttaaltaan tasainen. Keskitunnelia ei nelivedosta edes löydy, sillä moottoreita voi olla enemmän kuin yksi. Takakontti on syvä, ja sen lattiasta nousee tarvittaessa lisäistuimia. Kyytiin mahtuu seitsemän.

Sähköautoa ei käynnistetä eikä sammuteta. Model S on aina päällä. Virtanappia tai moottorin esilämmitystä ei siis tarvita. Koska myöskään päästöjä ei ole, matkustamon lämmitystä ja ilmastointia voi käyttää vaikka tallissa. Tesla on aina verkossa, joten lämpöä voi säädellä kännykällä. Model S oppii käyttäjänsä ajoaikataulut ja lämmittää tai jäähdyttää matkustamon valmiiksi. Auto ei tärise eikä savua parkissa eikä liikennevaloissa, sillä tyhjäkäyntiä ei ole. Auto vain on: hiljaa mutta valmiina.

Ilman vaihteistoa vääntö on aina välitöntä ja katkotonta. Jarruenergian talteenotto taas on niin voimakasta, että auto jarruttaa kaasua nostamalla. Sähköauton voi ”tankata” öisin kotitöpselistä ja Teslan sähköautot lisäksi ilmaiseksi Superchargerilla, joista ensimmäiset avattiin Suomeen toukokuussa. Tutut huollot kuten hihnat, sytytystulpat ja öljyt jäävät historiaan. Määräaikaishuollon käsite muuttuu, koska sähköautossa harva asia likaantuu tai liikkuu ja Teslan ohjelmistopäivitykset tulevat langattomasti perille.

Moni autoilun itsestäänselvyyksistä liittyy siis polttomoottoriin eikä olekaan enää itsestään selvää. Yhdessä Teslakin on silti yhä konservatiivinen: Model S on ulkoisesti varsin perinteisen näköinen. Parin vuoden päässä vartovan massamalli Model 3:n on vihjattu aloittavan rohkeamman linjan. Vieläköhän silloin muut sovittelevat akkuja bensatankkiin?

UNSCII uudistaa

Merkkigrafiikan päivitys Unicode-aikaan

Unicodessa on monia palikkagrafiikan kannalta herkullisia merkkejä, mutta juuri kukaan ei käytä niitä. Unscii-fontti yrittää ratkaista ongelman.

Teksti ja kuvat: Ville-Matias Heikkilä

Pari vuotta sitten halusin selvittää, onko Unicode löytänyt tiensä perinteiseen merkkigrafiikkaan. Osoittautui, että hyvin vähäisesti. Kyse ei selvästikään ole kehitysvastarinnasta: esimerkiksi Ecman päätteenohjausstandardin mukaiset laajemmat väripaletit on omaksuttu niin MUDeihin kuin muutamiin suosittuihin merkkigrafiikkaeditoreihinkin. Palikkavalikoimien osalta on kuitenkin pitäydytty perinnemerkistöissä – miksi ihmeessä?

Tasalevyisten fonttien vertaileminen vastasi kysymykseen: mikäli fontissa on grafiikkamerkkejä, yleensä vain PC:stä tutut merkit on toteutettu jokseenkin kunnollisesti. Kyseessä oli selvä muna–kana-ongelma: ennen kuin Unicode-palikkagrafiikka pääsisi kehittymään, pitäisi olla kunnollinen referenssifontti, jollaista ei kuitenkaan synny olemassa olevan grafiikkaperinteen puuttuessa.

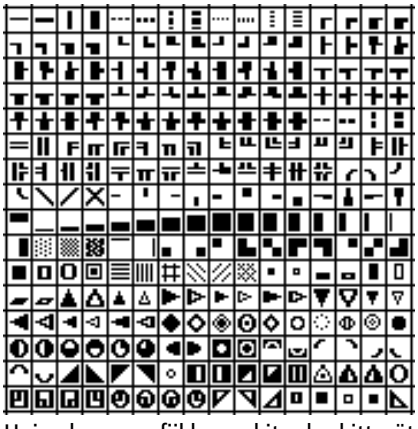
Merkkejä tutkimaan

Unscii-fontti syntyi halusta luoda 8×8 pikselin tasalevyinen bittikarttafontti, joka tukisi niin perinteistä kuin uuttakin merkkigrafiikkaa mahdollisimman hyvin. Nimen voi ajatella yhdistelmäksi termeistä Unicode, ANSI ja ASCII. Myöhemmin fontista alkoi syntyä tavallista pääte- ja ohjelmointikäyttöä varten myös 8×16-versio, josta olen mielinyt vaihtoehtoa iänikuiselle PC:n VGA-fontille.

Unicode-konsortio ei ota kantaa siihen, kuinka grafiikkamerkit pitäisi toteuttaa: niille on tarjottu paikat vain alaspäinyhteensopivuuden vuoksi. ”Oikean” toteutustavan löytämiseksi piti siis kaivaa esiin ne vanhat järjestelmäfontit, joilla näitä merkkejä on alun perin käytetty. Koska myös tavallisia kirjaimia, numeroita ja välimerkkejä on käytetty merkkigrafiikkaan, piti myös ne ottaa syyniin.

8×8 pikselin järjestelmäfontteja on karkeasti kahdenlaisia: ohutviivaisia, joissa viiva on yhden pikselin levyinen, ja paksuviivaisia, joissa paksuus on vaakasuunnassa kaksi pikseliä. Jälkimmäisiin kuuluvat niin Commodore 64:n, Amigan, Atari ST:n kuin PC:nkin oletusfontit, joten valinta oli tältä osin selvä. Otin vertailujoukkoon mukaan myös Atari XL:n ja Amstrad CPC:n ROM-fontit ja BBC Micron grafiikkatilafontin.

Paksuviivaiset 8×8-fontit ovat yllättävän samankaltaisia Ascii-alueensa osalta. Monet kirjaimet, numerot ja välimerkit ovat identtisiä liki koko



Unicoden grafiikkamerkit keskittyvät alueelle 2500-25FF, joka näyttää Unsciin 8×8-versiossa tältä.

vertailuryhmässä, joten niissä pääsi etenemään mainiosti enemmistöperiaatteella. Suuri osa viivapohjaisesta Ascii-grafiikasta on piirretty Amigan Topaz-fontille, joten joitakin sen piirteitä on suosittu. Esimerkiksi X:n ja Y:n viivat yhdistyvät luontevasti merkkeihin \, / ja |.

Varsinaisia grafiikkamerkkejä vertaillessani osoittautui, ettei monille niistä ole lainkaan Unicode-paikkaa! Fonttiprojektista tuli siis myös kartoitusprojekti, jossa kävin läpi kaikki mahdolliset grafiikkamerkkejä sisältävät perinnemerkistöt ja järjestelmäfontit.

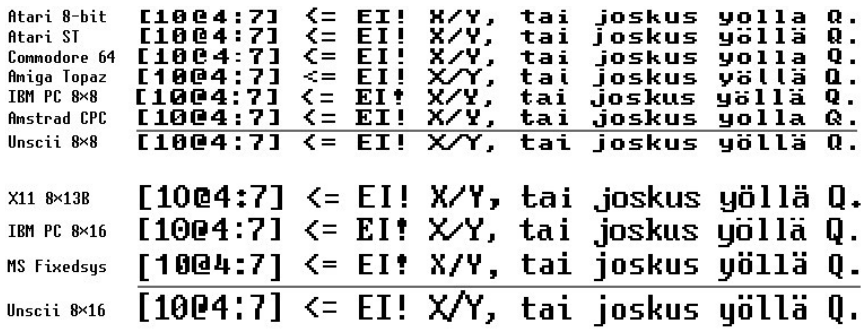
Suosituista merkistöistä esimerkiksi Petscii, Teksti-TV, Videotex ja jopa VT100 sisältävät useita käyttökelpoisia grafiikkamerkkejä, joita ei ole Unicodessa – tämä siitä huolimatta, että näitä tukevia laitteita on ollut käytössä miljoonittain. Uusia merkkejä löytyi lisäksi CPC:n, Atarin kotimikrojen, EACA Colour Genien, Mattel Aquariuksen, Sharp MZ:n ja TRS-80:n ROM-fonteista sekä Applen Sabine10-merkistöstä. Päätin sijoittaa nämä merkit Unsciissa PUA-alueelle (Private Use Area), eli koodipaikasta U+E000 alkaen.

Kaksi eri leveyttä

Monet Unicoden grafiikkamerkeistä ovat peräisin kiinalais-japanilais-korealaisista merkistöistä (CJK), joten otin tarkasteluun myös japanilaisen NEC PC:n ja muutaman itäisen MSX-koneen ROM-fontit. Näissä koneissa merkeillä on kaksi mahdollista leveyttä: puolileveä (8 pikseliä) ja täysleveä (16 pikseliä). Esimerkiksi kiinalaisperäiset sanamerkit ovat täysleveitä.

Länsimaiset pääteohjelmat käyttävät merkin leveyden määrittämiseen yleensä `wcwidth()`-funktiota. Funktio ilmoittaa itsenäisen merkin leveydeksi aina yhden solun, ellei sitä ole erikseen määritelty täysleveäksi itäaasialaismerkiksi, jolloin leveys on kaksi solua. Idässä käytäntö on kuitenkin erilainen: merkin leveys riippuu siitä, käytetäänkö merkin siirtokoodauksessa yhtä vai kahta tavua. Suurin osa grafiikkamerkeistä on kaksitavuisia, joten täydellisen fontin pitäisi pystyä esittämään ne sekä yhden että kahden solun kokoisina.

Monet täysleveät grafiikkamerkit pystyy esittämään tyydyttävästi kahden puolileveän Unicode-merkin yhdistelmänä, mutta esimerkiksi taiwanilaisessa BBS-grafiikassa suosittuja kulmakolmioita ei. Niinpä tärkeimmistä grafiikkamerkeistä on Unsciissa tarjolla myös vasen ja oikea puolisko erikseen. Puolitetut merkit helpottavat myös mittasuhdemuunnoksissa, esimerkiksi 40×25 merkin ruudulle piirretyn kuvan venyttämisessä samanmuotoiseen 80×25 merkin ikkunaan.



Unscii muistuttaa tekstimerkkiensä osalta mikrotietokonemaailman järjestelmäfontteja.

Esikuvia ja ryöstöretkiä

Unsciin 8×16-versio on muodostettu 8×8-fontin pohjalta siten, että grafiikkamerkkien mitat on pidetty solun suhteen mahdollisimman ennallaan mutta tekstiglyyfejä on mataloitettu ja kaunistettu. Esikuvana ovat olleet etenkin PC:n 8×16-kokoinen VGA-fontti ja Windowsin 8×15-kokoinen Fixedsys, jotka ovat läheistä sukua paksuviivaisille 8×8-fonteille. Muita esikuvia ovat olleet X-ikkunointijärjestelmän lihavoidut 8×13- ja 9×15-fontit sekä VT420-päätteen 10×16-fontti. Tarvittaessa mallia on otettu myös suosituista tasalevyisistä vektorifonteista, joita ovat DejaVu Sans Mono, Lucida Console, Monaco ja Inconsolata.

Unscii on toteutettu piirtämällä sen glyyfit tekstitiedostoihin pistettä ja risuaitaa käyttäen. Tekstitiedostoissa on lisäksi fonttien Unicode-paikat sekä erilaisia ohjauskäskyjä, joita on projektin edetessä määritelty lisää tarpeen mukaan. Tekstitiedostot käännetään BDF-muotoon Perl-skripteillä ja BDF-muodosta tuotetaan PCF- ja TTF-versiot erillisillä työkaluilla.

Unsciin Unicode-kattavuus ei ole sellaisenaan kovinkaan hyvä, minkä huomaa esimerkiksi jonkun kopioidessa IRC:hen kiinankielistä tekstiä tai foneettisia ääntämisohjeita. Tämän vuoksi 8×16-Unsciista on myös versio, johon on haettu puuttuvat glyyfit skriptin avulla GNU Unifontista.

Unifont pyrkii olemaan mahdollisimman kattava bittikarttafontti. Sen glyyfit ovat oikeankokoiset – 8×16 ja 16×16 pikseliä – mutta ne eivät ohutviivaisina istu Unsciin tyyliin. Skripti asemoi glyyfit uusiksi ja skaalaa ne tarvittaessa wcwidthin mukaisiksi, mutta viivanleveyteen ei kosketa. Fixedsysin harrastelijavetoinen Unicode-versio Fixedsys Excelsior olisi tyyliltään yhteensopivampi, mutta sen lisenssi on turhan epämääräinen.

Koeajo

Unsciita on kehitysvaiheessa testattu niin grafiikassa kuin normaalissa päätekäytössäkin: ohjelmoinnissa, tekstinmuokkauksessa ja irkkauksessa. Muutamia typografisia ongelmia on korjattu: toisiinsa helposti sekoittuvien merkkien erottuvuutta on parannettu etenkin ohjelmointikäyttöä ajatellen ja tekstijäljestä on tehty tasaisempaa pidentämällä kapeiden I-, i- ja l-kirjainten päätteitä.

Olemassa olevat merkkigrafiikkaeditorit eivät taivu kovinkaan helposti Unicodeen eivätkä tekstieditorit merkkigrafiikkaan. Tein palikkagrafiikkakokeiluja varten aluksi joitakin makroja tekstieditoriin ja myöhemmin puhtaalta pöydältä oman grafiikkaeditorin. Grafiikan piirtelyyn sain innoketta myös Simulaatio-demotapahtumaan työstämäni merkkigrafiikkademon kautta. Demoa varten on tosin koottu merkeistä suppeampi, 256 merkin valikoima.

Tätä kirjoitettaessa Unscii on vaiheessa, jossa sen kehtaa jo laittaa julkisesti ladattavaksi. Käy siis lataamassa fontti vaikkapa Skrollin sivulta http://skrolli.fi/2015.2 ja kerro kokemuksistasi! Koska seuraava etappi tulee olemaan jonkinlaisen grafiikkakilpailun järjestäminen, ovat myös merkkigrafiikkapiirrokset hyvin tervetulleita – niin värittömät kuin värillisetkin!

Neo Geo MVS:n pelimoduuli (AES-moduuli oli saman kokoinen). Mittakaavana NES:n peli.

Ei näin! Siis Noinko Kallis?

SNK teki menestyneitä kolikkopelejä ja siinä sivussa äärimmäisen pelikonsolin. Etenkin heille, jotka sattuivat olemaan miljonäärejä.

Teksti: Mikko Heinonen

Kuvat: Mikko Heinonen, Manu Pärssinen, Wikimedia Commons

Shin Nihon Kikaku -niminen yhtiö syntyi 1970-luvun lopussa Japanin kasvaville kolikkopelimarkkinoille. Sen ensimmäiset pelit olivat useimpien aikalaistensa tavoin teknisesti vaatimattomia, eivätkä useimmat niistä kohonneet suuriksi menestyksiksi. Laatu parani hiljakseen, mutta vasta vuoden 1986 Ikari Warriors saavutti niin suurta suosiota, että se käännettiin liki kaikille kodeissa käytetyille pelialustoille.

Ikarin myötä SNK alkoi varovasti kiinnostua myös kotilaitteiden markkinoista. Se hankki muun muassa oikeudet kehittää pelejä Nintendo Entertainment Systemille. Tämä panostus ei lopulta tuottanut kuin kaksi peliä, mutta se oli epäilemättä eräänlaista alkusoittoa seuraavalle suurelle seikkailulle: oman pelikonsolin kehittämiselle.

Monivideojärjestelmä syntyy

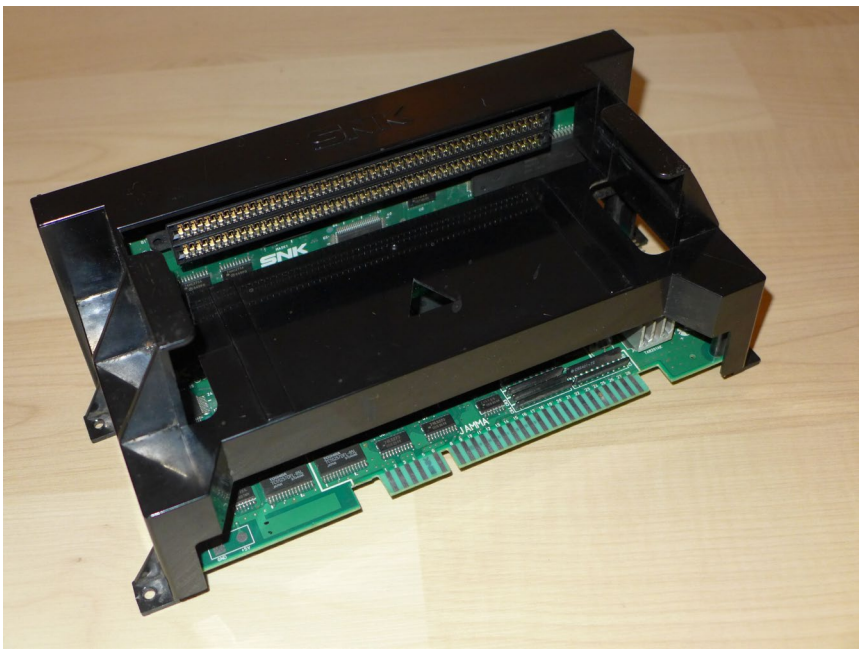
Eräs pelihallien perusongelma oli, että kolikkopelien uusiminen oli kallista. Koska jokainen pelilevy sisälsi myös kaiken sen pyörittämiseen tarvittavan elektroniikan, oli pelin vaihtaminen toiseen usein huomattava investointi. Pelaajat kuitenkin osasivat vaatia uutuuksia ja vaihtelua pelivalikoimaan.

Jo varhaisessa vaiheessa tähän markkinarakoon oli tarjoutunut monenlaista yrittäjää, kuten Data Eastin Cassette System. Se latasi pelin pieneltä mikrokasetilta, kun kabinetti käynnistettiin. Kasetit olivat hitaita ja mekanismi epäluotettava, joten järjestelmä ei yleistynyt. Kukaan muukaan ei ollut saanut ratkaistua ongelmaa tavalla, joka olisi todella houkutellut pelinkehittäjiä.

SNK:n vuonna 1990 esitelty tarjokas oli nimeltään Neo Geo Multi-Video System. Käytännössä se oli kuin turboahdettu pelikonsoli, joka asennettiin kolikkopelin kabinettiin. Peliä saattoi vaihtaa yksinkertaisesti vaihtamalla ROM-moduulin toiseen, ja tarjolla oli myös versioita, joihin sopi useampi peli kerralla. Rauta oli tuon ajan mittapuulla erittäin järeää: suorittimina toimivat rinnakkain Motorola MC68000 ja Zilog Z80, grafiikkapiiri puski ruudulle 380 spriteä 4096 värillä, äänet soivat 15 kanavan voimin ja moduuleilla oli muistitilaa jopa 330 megabittiä, myöhemmin pankitettuna vieläkin enemmän.

MVS olikin välitön menestys. Idea oli pohjimmiltaan yksinkertainen, mutta kukaan ei ollut aiemmin ympännyt sitä näin kyvykkääseen rau-

Neo Geo AES:n mukana toimitettiin myös laadukas ohjain.



Neo Geo MVS -emolevy.

taan. Alle kymmenessä vuodessa käytännössä koko kolikkopeliteollisuus siirtyisi käyttämään vastaavia järjestelmiä. Myös itse Neo Geolle julkaistaisiin pelejä vielä yli vuosikymmen myöhemmin. Alalla, joka on aina kulkenut tekniikan eturintamassa, tämä on hämmästyttävä suoritus. Mutta SNK tekisi silti konkurssin.

Liiankin edistynyttä viihdettä

Neo Geo MVS oli käytännössä eräänlainen pelikonsoli ilman kuoria. Siksi ei ollut mikään valtava loikkaus tehdä siitä ihan oikeakin pelikonsoli: Neo Geo Advanced Entertainment System. Se oli aikansa tuotteeksi todella edistynyt laite ja tarjosi muun muassa muistikorttipaikan pelin tallennukseen. Pelinsä saattoi jopa tallentaa pelihallissa ja jatkaa sitten kotona tai päinvastoin.

Huipputekniikalla on luonnollisesti huippuhinta, joten AES:stä tuli varsin kallis. SNK tarjosi sitä ensin lähinnä vuokrakäyttöön ja hotelleihin, mutta päätyi lopulta asiakaspalautteen perusteella julkaisemaan koneen myös myyntiin loppuvuodesta 1990.

Konsolin korkeahko hinta oli lopulta pienempi pulma kuin se, että pelit maksoivat aivan poskettomasti. Kun Neo Geo lopulta saatiin Suomeen, konsolin saattoi saada reilulla puolellatoista tuhannella markalla (nykyrahassa inflaatio huomioiden noin 350 €), mikä ei sinällään ollut kohtuutonta. Muistikapasiteetiltaan suurimmat pelit maksoivat kuitenkin lähes yhtä paljon kuin itse konsoli, eli jopa kolmin- tai nelinkertaisesti sen, mihin Nintendon ja Segan asiakkaat olivat tottuneet. Eipä silti, itse moduulit olivat myös paljon suurempia kuin kilpailijoilla.

Hulppea hinta oli seurausta siitä, että pelit olivat itse asiassa täsmälleen samoja kuin MVS:llä. Ne vaativat esimerkiksi paljon kalliita muistipiirejä. Kolikkopeleissä moduulien tuotantokustannukset eivät olleet suurikaan haitta, sillä operoinnissa hyvä peli ansaitsi kyllä hintansa takaisin. Kuluttajatuotetta oli kuitenkin varsin hankalaa markkinoida siihen hintaan, joka AES-moduuleihin lätkäistiin.

Oli muutenkin vaikeaa tarjota käytännössä samaa tuotetta kahteen eri hintaan: halvemmalla kuluttajille ja kalliimmalla ammattikäyttöön. Vaarana oli, että kekseliäät operaattorit hankkisivat AES:n moduuleita ja peluuttaisivat niitä rahaa vastaan. Tämän vuoksi koti- ja ammattimoduulien nastajärjestykset sotkettiin niin, etteivät ne sopineet ristiin.

Hopeakiekkoja ja jonglööreita

Oli melko selvää, että Neo Geo AES:stä ei voisi koskaan tulla jättimenestystä, jos sen pelit maksaisivat konsolin verran. Siinä missä esimerkiksi Sega Megadriveä myytiin 40 miljoonaa kappaletta, Neo Geon myynti ei koskaan rikkonut edes miljoonan rajaa.

Vuonna 1994 SNK yritti uudelleen Neo Geo CD:llä. Pelit oli siirretty halvemmalle tallennusvälineelle, josta ne ladattiin konsolin muistiin. Ratkaisu laski pelien hinnan murto-osaan entisestä.

CD-ROMin käytöstä seurasi kuitenkin uusi ongelma, sillä sen lukunopeus ei päätä huimannut. Asema toimi 1X-nopeudella eli latasi parhaimmillaan 150 kilotavua sekunnissa. Tyypillinen kerralla muistiin ladattava osio saattoi olla liki 10 megatavua, joten pelaaja sai tottua odottelemaan. Latauspalkin päällä jonglööraava apina lakkasi huvittamasta varsin pian. Edistyneempi CDZ-versio oli sentään varustettu tuplanopeuksisella asemalla.

Myös CD-version myynti jäi vaatimattomaksi: lähteissä puhutaan alle 600 000 kappaleesta. Teknisten seikkojen lisäksi

Neo Geo CD:stä ilmestyi kolme eri mallia, tässä niistä toinen eli ”toploader”.

asiaan vaikutti varmasti myös se, että Neo Geon pelit olivat käytännössä ainoastaan kolikkopelikäännöksiä. Teknisestä etevyydestä huolimatta koneelle ei koskaan ilmestynyt mitään kovin syvällistä pelattavaa, sillä pieni konekanta ei houkutellut julkaisijoita.

Neo Geo jäi alkuperäisen maineensa vangiksi ja todellisten hardcore-pelaajien herkuksi. Heitä tosin ilahdutti se, että varsin pian markkinoille saapui sovittimia, joilla AES:llä saattoi pelata MVS:n pelejä. Niitä taas vapautui operointikäytöstä myyntiin lopulta paljon halvemmalla kuin kotiin tarkoitetut pelit olivat koskaan olleet, mikä teki koko nastajärjestyskikkailusta melko tarpeetonta.

Neo Geo Pocket Color.

No sopisiko se taskuun?

Kolmas yritys nähtiin vuonna 1998, kun Neo Geo Pocket ilmestyi. Nintendo Game Boyn menestyksestä innostuneena myös SNK päätti kokeilla siipiensä kantavuutta taskupelien markkinoilla. Tavoilleen uskollisesti NGP:stäkin tehtiin laadukkaan tuntuinen laite: etenkin mikrokytkimillä toteutettu ohjaussauva on edelleen yksi parhaista lajissaan. Vanhan Neo Geon kanssa laitteella ei tosin ollut tekemistä kuin nimen ja tiettyjen pelisarjojen verran.

Neo Geo Pocketin myynti käynnistyi kovin vaatimattomasti. Syykin oli selvä: Game Boysta oli juuri julkaistu uusi Color-versio, kun taas NGP oli varustettu mustavalkonäytöllä. SNK teki nopean korjausliikkeen, lopetti Pocketin myynnin ja julkaisi jo seuraavana vuonna Neo Geo Pocket Colorin. Se oli muuten sama laite, mutta pystyi tuottamaan 2,7 tuuman näytölleen peräti 146 väriä – kolminkertaisesti Game Boy Coloriin nähden.

NGPC kävikin heti kaupaksi aivan eri tavalla. Se oli sitten Sega Game Gearin ensimmäinen konsoli, joka onnistui tekemään minkäänlaista lovea Nintendon monopoliin. Hyvä tekniikka, liki 40 tunnin toiminta-aika yksillä paristoilla ja SNK:n laadukkaat arcade-pelit vetosivat ostajiin aluksi, mutta sitten veto alkoi hiipua.

Päällimmäinen syy oli varmasti se, että Pokémon-villitys ampui Nintendon ja Game Boyn aivan uudelle kasvu-uralle. SNK:lla ei ollut tarjota mitään vastaavaa, eikä se onnistunut houkuttelemaan järjestelmälleen ulkopuolisia kehittäjiä. Kuvaavaa oli, että NGPC:lle myytiin kaapelia, jolla sen saattoi yhdistää toiseen menestyjään – Sega Dreamcastiin. Kahden miljoonan kappaleen myynti oli SNK:lle paljon, mutta kilpailijaan nähden liki olematonta.

Tällä välin itse yhtiö oli joutunut pahoihin vaikeuksiin. Neo Geo MVS oli jo vanhentunut, sen seuraaja Hyper Neo Geo 64 ei lähtenyt koskaan nousukiitoon ja rahat olivat lopussa. Pachinko-koneiden valmistaja Aruze osti SNK:n vuonna 2000 ja lähes ensi töikseen lopetti sen toiminnot Japanin ulkopuolella. Hämmentävästi jopa myymättömät, länteen tarkoitetut Neo Geo Pocket Colorit ja niiden pelit kuljetettiin takaisin Japaniin ja paketoitiin siellä uudelleen paikallisille markkinoille. Tämä ei laivaa kääntänyt, vaan SNK kaatui konkurssiin loppuvuodesta 2001.

Pelaamme enemmän

SNK:n perustaja Eikichi Kawasaki näki jo ennen konkurssia, että laiva ajaa pian kiville. Hän perustikin vähin äänin Playmore Corporationin, jolle SNK:n raadon arvokkaat osat siirtyivät väistämättömän tapahduttua. Vuodesta 2003 yritys on toiminut nimellä SNK Playmore, hallinnoiden lähinnä SNK:n arvokkaimpia pelisarjoja kuten Metal Slug, King of Fighters ja Samurai Shodown sekä niiden käännöksiä eri alustoille.

Vuonna 2012 SNK Playmore antoi Tommo-nimiselle valmistajalle lisenssin tehdä Neo Geosta uusintaversion nimellä Neo Geo X. Linux-pohjainen käsikonsoli oli mahdollista liittää Neo Geo AES:n näköiseen telakkaan, jolla se yhdistyi televisioon. Mukana oli varsin laadukkaan tuntuinen ohjain ja paketissa toimitettiin nippu vanhoja Neo Geo -pelejä muistikortilla. Myös hinta oli varmuuden vuoksi ruuvattu yläkanttiin.

Valitettavasti etenkään ensimmäisen valmistuserän laiteohjelmisto ei ollut totuttua Neo Geo -laatua, mikä suututti monet konsolin vanhat fanit. Pelien äänet pätkivät ja kuvanlaatu oli suttuinen. Lisänurinaa aiheutti se, että kaivatun laiteohjelmistopäivityksen sai ainoastaan ostamalla uuden pelipaketin. Jollain tavalla tämä tosin sopikin kuvioon, sillä Neo Geolla ei ole oikein koskaan pelattu ilmaiseksi.

Sori, Näin Kävi

Ei näin! -artikkelisarjan mittapuulla SNK ei ole sieltä epäonnistuneimmasta päästä. Firma on tavallaan edelleen olemassa ja sen konsolit ja pelit nauttivat ansaittua kulttimainetta. Tunnetta hukatusta potentiaalista on silti vaikea välttää. Neo Geo olisi voinut olla paljon enemmänkin kuin vain pelihallien hirmu ja hc-pelaajien salaseuran jäsenkirja. Tekninen tinkimättömyys kuitenkin teki laitteesta liian arvokkaan massamarkkinoille. Pienet myönnytykset massatuotannolle olisivat ehkä varmistaneet suuremman suosion ja sitä kautta tuotekehityksen paremman jatkuvuuden.

SUOMEN PELIMUSEO

Suomalainen pelikulttuuri ansaitsee museon!

TEHDÄÄN SUOMEN PELIMUSEO YHDESSÄ.

Tue meitä ostamalla lippu ennakkoon ja pääset tutustumaan museoon jo ennen avajaisia.

OSALLISTU JOUKKORAHOITUKSEEN:

www.mesenaatti.me/suomenpelimuseo

Museokeskus Vapriikkiin Tampereelle avataan loppiaisena 2017 Suomen pelimuseo, joka kertoo suomalaisen pelaamisen monikymmenvuotisen historian sen ansaitsemissa puitteissa.

Suomen pelimuseo-hanke on syntynyt Vapriikissa sijaitsevan Mediamuseo Rupriikin, Pelikonepeijoonien ja Tampereen yliopiston yhteistyöprojektina.

Upeat ja unohdetut Unix-koneet

Vanha kotitietotekniikka on monille läheistä. Vähemmän tunnettuja ovat vanhat ammattilaiskoneet – kuten Unix-työasemat ja -palvelimet.

Teksti: Ville-Matias Heikkilä

Kuvat: Mikko Torvinen, Wikimedia Commons-käyttäjät Shieldforyoureyes, Zymos, Brian Pitts, Napoli Roma, Thomas Kaiser, Fluff, Modano, Thomas Schanz, allaboutapple.com

Unix-työasemat ovat työpöytäkokoluokan koneita, jotka on suunniteltu ajamaan jotain Unix-sukuista käyttöjärjestelmää. Niiden tausta on kuitenkin toisenlainen. Siinä, missä PC:t ovat alkujaan kotitietokoneita, jotka laajentuivat ammattikäyttöön, ovat Unix-työasemat pikemminkin isojen koneiden kutistettuja versioita.

Työasemakoneita on käytetty etenkin sellaisissa raskaissa ammattitehtävissä, joihin ”varsinaisten” mikrotietokoneiden rahkeet eivät ole riittäneet. Tällaisia ovat olleet esimerkiksi teollisuussuunnittelu, 3D-grafiikka ja tieteellinen tutkimus. Työasemia on usein käytetty myös palvelimina, ja useimmilla valmistajilla on ollut valikoimissaan myös työasemamallien kanssa yhteensopivia palvelinmalleja. Myöhemmin valmistajat alkoivat keskittyä yksinomaan palvelimiin.

Mikron ja minin ristisiitokset

Tietokoneen käyttö oli 1970-luvulla enimmäkseen ositusajoa. Henkilökohtaisen tietokoneen idea nosti kuitenkin päätään kahdelta suunnalta: toiset halusivat rakentaa halvoista mikropiireistä jotain juuri ja juuri tietokoneeksi kelpaavaa, toiset taas unelmoivat laitteista, joissa ”täysiverisen” tietokoneen resurssit keskitettäisiin yhden käyttäjän palvelemiseen. Varhainen esimerkki jälkimmäisentyyppisestä koneesta on *Xerox Alto* (1973), joka tunnetaan graafisen käyttöliittymän ja Ethernet-lähiverkon pioneerina.

Unix suunniteltiin alkujaan ositusajokäyttöjärjestelmäksi, jota käytettiin tekstipäätteiden yli Digitalin *PDP-11*- ja *VAX*-minitietokoneilta. 1980-luvun alussa alkoi markkinoille kuitenkin tulla yrityksiä, jotka kehittivät Unix-pohjaisia yhden käyttäjän koneita: *Apollo Computer*, *Sun Microsystems*, *Silicon Graphics*. Myöhemmin myös *Digitalin* ja *IBM:n* kaltaiset konkarit hyppäsivät kelkkaan. Alkujaan avoimesti kehitetty Unix-käyttöjärjestelmä alkoi tällöin kaupallistua ja samalla jakautua valmistajakohtaisiin variantteihin.

80-luvun työasemat perustuvat yleensä Motorolan 68000-sarjan suorittimeen, joka muistuttaa käskykannaltaan edellämainittuja Digitalin minitietokoneita. 1990-luvulla käytössä olivat 32- ja 64-bittiset RISC-suorittimet, jotka kukin laitevalmistaja yleensä kehitti ja valmisti itse. Pitkälle viety valmistajakohtaisuus johti myös yhteensopivuusongelmiin eri Unix-varianttien välillä.

Vuosituhannen vaihteen tienoilla PC-koneet olivat ottaneet Unix-työasemat jo monin tavoin kiinni. Useilla valmistajilla oli tämän vuoksi talousvaikeuksia, ja ne rupesivatkin keskittymään superlaskenta- ja palvelinrautaan. Työasemien tuotanto lopetettiin, tai ne vaihdettiin PC-pohjaisiksi. Käytöstä poistettuja työasemakoneita rupesi enenevässä määrin päätymään harrastajille.

Mitä sillä voi tehdä?

Unix-työasemat eivät ole olleet aikoihin realistisesti varteenotettava vaihtoehto PC-raudalle. Niissä on kuitenkin edelleen oma hohtonsa ja vanhan ajan ammattilaisuskottavuutensa. Ne poikkeavat viehättävillä ja omituisilla tavoilla vaikkapa Linuxia ajavista PC-koneista mutta ovat kuitenkin sen verran lähellä niitä, että käyttökokemus on samankaltainen. Ne ovat myös fyysisesti sen verran pieniä, ettei niitä harrastavan tarvitse hankkia varastohalleja laitteiston säilömiseen.

Unix-koneita on harrastettu etenkin rakentamalla niistä palvelimia ja verkkoja. Eri laitteistot ja käyttöjärjestelmät erilaisine hienouksineen ja kommervenkkeineen tuovat touhuun sellaista mielenkiintoisuutta, mitä pelkästä

Linux-tunkkauksesta ei saa. Laitteilla voi kuitenkin toki tehdä muutakin. Nyrkkisääntönä voi pitää, että jos jokin ohjelma toimii Raspberry Pi'n Linuxilla, niin sen saa toimimaan myös 90-lukulaisessa työasemakoneessa.

Monista perinteisistä Unix-käyttöjärjestelmistä ilmestyy edelleen uusia päivitysversioita, käytetäänhän niitä edelleen tärkeissä palvelinkoneissa. Aidon ohjelmiston löytäminen saattaa joskus vaatia hieman tuuria, ja jos sitä ei ole, voi konetta usein testata myös Linuxin tai NetBSD:n kaltaisilla vapailla unixeilla.

Omalaatuiset ongelmat

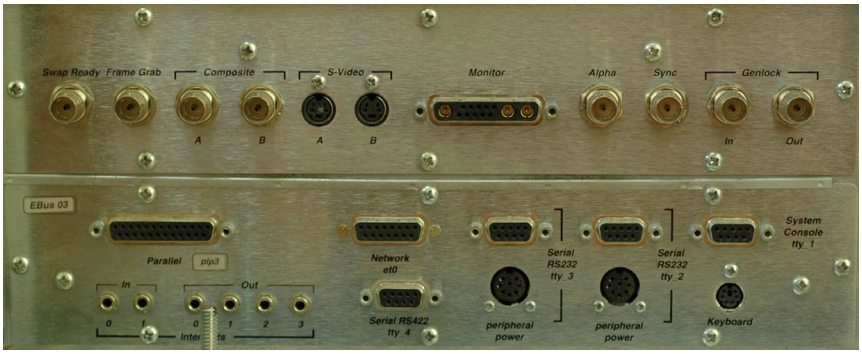
Kun Unix-kone on haalittu kotiin, seuraa yleensä joukko ongelmia. Näyttö liitetään usein *13w3*- tai koaksiaali-rgb-liittimellä eikä millään PC:stä tutulla. Ongelman voi ratkaista joko juottamalla tai tyytymällä sarjaporttipäätteeseen. Näppäimistöt ja hiiret ovat yleensä täysin vierasta tavaraa. Kiintolevyt, optiset asemat ja jopa korppuasemat ovat useimmiten SCSI-pohjaisia kaikissa paitsi aivan peeseemäisimmissä halpismalleissa.

Jos koneen levy on tyhjennetty tai poistettu, pääsee harrastaja ensi töikseen etsimään käyttöjärjestelmän ja asentamaan sen. Asentamisessa voi olla omat konstikkuutensa – esimerkiksi asennus-CD ei välttämättä boottaa ellei CD-aseman blokkikoko ole oikea. Asennuksen voi yleensä käynnistää myös verkon yli TFTP:llä, mutta se vaatii usein toisen koneen, joka ajaa samaa käyttöjärjestelmää.

Jos tarjolla on Gnu-ohjelmistopaketteja, ne kannattaa asentaa. Kaupallisten unix-kalujen toiminta eroaa usein jonkin verran Gnusta, mikä aiheuttaa yhteensopivuusongelmia. Valmistajan oma C-kääntäjä osaa yleensä tehdä matalan tason optimoinnit paremmin kohdesuorittimella, mutta GCC on yhteensopivampi ja päihittää korkeamman tason optimoinneissa.

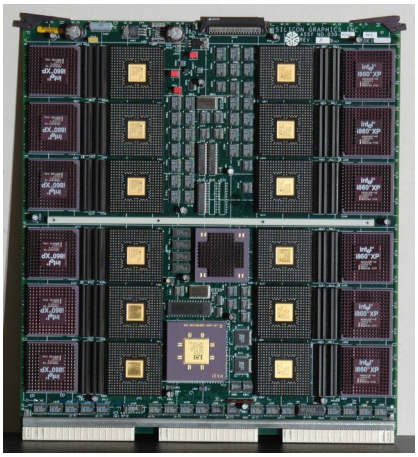
Kaupallisessa Unix-maailmassa saattaa usein törmätä *open*-sanaan. Se kuitenkin tarkoittaa yleensä valmistajariippumattomuutta eikä vaikkapa koodin julkisuutta. Esimerkiksi *CDE*-työpöytä ja *Motif*-kirjasto ovat X-ikkunointijärjestelmän "avoimia" lisäosia, jotka eivät koskaan yleistyneet Linux-puolella kaupallisuutensa vuoksi.

Unix-koneet ovat kaiken kaikkiaan varsin mielenkiintoisia vekottimia. Jos siis romun seassa tulee vastaan vanha Unix-työasema, -palvelin tai vaikkapa pääte, kannattaa se ehdottomasti ottaa talteen – ainakin jos vaihtoehtona on vanha kuluttaja-PC-klooni.



SGI Onyxin liitäntöjä.

Sunin 64-bittinen UltraSPARC 1 -suoritin.


SGI Onyxin Geometry Engine GE10-näytönohjain.


Erään harrastajan Unix-työasemakokoelma.

Päätteet

Unix-koneita on aina käytetty myös monen käyttäjän palvelimina, joten Unix-rautaa etsiessä törmää varmasti ennen pitkää myös päätteisiin. Päätteitä on karkeasti kahdenlaisia: RS232-liitäntäisiä tekstipohjaisia "tyhmiä päätteitä" ja Ethernet-verkon yli toimivia graafisia X-päätteitä. Päätteet koostuvat yleensä kuvaputkinäytöstä, samoihin kuoriin rakennetusta päätelogiikasta ja erillisistä syöttölaitteista, ja ne saa varsin pienellä vaivalla käyttöön niin Unix-koneiden kuin nykyisten Linux-PC:idenkin jatkeeksi.

Ehkä tunnetuin tekstipäätteiden valmistaja on Digital, joka kehitti standardin asemaan nousseen *VT100:n*. Vaikka käytännössä kaikki nykyisin käytetyt pääte-emulaattorit ovatkin VT100-yhteensopivia, on eri valmistajien päätteissä hyvin paljon vaihtelua ohjauskoodeissa. *Screenin* kaltaiset virtuaalipääteohjelmistot paikkaavat kuitenkin yhteensopivuusongelmat.

X-ikkunointijärjestelmä on alusta asti käyttänyt asiakas-palvelin-mallia, joten periaatteessa mitä tahansa X-ohjelmaa saa käytettyä X-päätteellä aina nykyisiä webbiselaimia myöten. Ongelmia voivat kuitenkin tuottaa vanhojen päätteiden rajalliset väripaletit ja monien uusien ohjelmien hinku OpenGL-rajapintoihin.

Sun Microsystems

Sun Microsystems perustettiin 1982, ja 68000-pohjainen *Sun-1* julkaistiin samana vuonna. Vuonna 1987 julkistetussa *Sun-4*:ssä yhtiö siirtyi omaan *SPARC*-suoritinperheeseensä.

32-bittinen *SPARCstation 1* oli vuonna 1990 Jyrki J. J. Kasvin "unelmamikro", joka jäi sadan tonnin hinnastaan johtuen pelkäksi haaveeksi. Kun Skrollin päätoimittaja kymmenen vuotta myöhemmin hankki kyseisen laitteen, pyydettiin siitä enää 150 markkaa.

Sunin Unix-variantti on nimeltään *SunOS* tai *Solaris*. Solariksella ja Sunin raudalla yleensäkin on erityisen hyvä maine palvelinkäytössä, jossa se on osannut hyvin hyödyntää isoja suoritinmääriä. Solaris on saatavilla myös X86:lle ja on käytettävissä ilmaiseksi ei-kaupalliseen käyttöön.

Sun on rampauttanut tahallaan halvempien työasemiensa ominaisuuksia, jotta kalliimmat koneet vaikuttaisivat riittävän paljon paremmilta. Esimerkiksi halvoissa näytönohjaimissa on käytetty PC:stä tuttuja piirejä, joiden kiihdytysominaisuuksia ei kuitenkaan tueta lainkaan. Myös IDE-levyohjainten ajurit ovat täysin suoritinvetoiset.

2000-luvulla varsin yleisiä työpöydillä ovat olleet graafiset *Sunray*-päätteet, joiden hienoutena on mahdollisuus istunnon "siirtämiseen" päätteeltä toiselle id-kortin avulla. Sunraylla voi etäkäyttää niin X-ikkunointia kuin Windowsiakin.

Sunin osti vuonna 2009 tietokantayhtiö *Oracle*. SPARC-suorittimien ja niitä käyttävien supertietokoneiden kehitystyö jatkuu edelleen.

Silicon Graphics

Silicon Graphics perustettiin Sunin tapaan 1982. Sen ensimmäinen tuote oli *IRIS 1000* -grafiikkapääte, mutta *IRIS 2000* ja *3000* olivat jo täysverisiä Unix-työasemia. Firma siirtyi 68000:sta *MIPS*-arkkitehtuuriin vuonna 1986 ja osti MIPSin kokonaan itselleen 1992.

SGI on profiloitunut alusta asti 3D-grafiikkaan, ja etenkin 90-luvun Hollywood-tietokonegrafiikka antoi sille nimeä ja näkyvyyttä. Jurassic Park -elokuvan surullisenkuuluisessa Unix-hakkerointikohtauksessa esiintyy 64-bittinen *Crimson*-työasema, joka ajaa kolmiulotteista *fsn*-tiedostonhallintasovellusta.

SGI on 3D-kiihdytyslaitteiston pioneereja. Indigo-työasemassa näytönohjauslaitteisto voi olla suurempi kuin koneen varsinainen emolevy. SGI:n kehittämiä ovat myös *Playstation 1:n* ja *Nintendo 64:n* näytönohjauslaitteistot, ja molemmat konsolit käyttävät SGI-työasemien tapaan MIPS-suorittimia. Edelleen teollisuusstandardin asemassa oleva *OpenGL*-grafiikkarajapinta syntyi alun perin vuonna 1992 *IRIS GL* -grafiikkakirjaston pohjalta.

Suosituimpiin SGI:n työasemiin kuuluu vuonna 1996 julkistettu "edullinen" *O2*, jonka yläpään malleissa on muun muassa sisäänrakennettu kamera. Yhtenäismuistiarkkitehtuurin ansiosta koko enimmillään gigatavun muistikapasiteetti on käytettävissä näytönohjaukseen. O2 oli parina vuonna Suomen Assemblyilläkin kilpailupalkintona.

IRIX-käyttöjärjestelmällä ja sen myötä myös SGI-raudalla on huono maine palvelinkäytössä kilpailijaansa Suniin nähden reikäisyytensä ja epävakautensa vuoksi.

Silicon Graphics International, joka on lakiteknisesti eri firma kuin vuonna 2009 konkurssin tehnyt alkuperäinen SGI, keskittyy X86-pohjaisiin supertietokoneisiin ja palvelimiin. MIPS ja IRIX hylättiin jo aikapäiviä sitten.

Digital Equipment Corporation

Digital Equipment Corporation valmisti tietokonekomponentteja jo 50-luvulla. Unix kehitettiin alkujaan DEC:n *PDP-8*- ja *PDP-11*-minitietokoneilla, mistä se käännettiin myös 32-bittisille *VAX*-koneille 70-luvun lopulla.

DEC:n ensimmäisenä Unix-työasemana voidaan pitää VAX-pohjaista *VAXstationia*, joka tuli markkinoille 1984. Vaikka VAXin oletuskäyttöjärjestelmä onkin firman oma *VMS*, joka ei ole mitään sukua Unixille, oli sille alusta asti tarjolla myös DEC:n oma Unix-variantti *Ultrix*. Ultrix toimii VAXin lisäksi myös PDP-11:ssä ja MIPS-pohjaisissa *DECstation*-työasemissa.

Vuonna 1992 DEC otti käyttöön oman 64-bittisen RISC-arkkitehtuurinsa, *Alphan*. Alpha on tunnettu laskentanopeudestaan, ja myös itse Cray valitsi sen 90-luvun supertietokoneisiinsa. Alphan oli jossain vaiheessa tarkoitus syrjäyttää myös PC:n X86, ja harrastajien käsiin päätyneet Alpha-laitteet ovatkin usein nimenomaan Alpha-suoritinta käyttäviä PC-emolevyjä.

Alphalla ajettava Unix-variantti oli aluksi nimeltään *OSF/1*, josta se vaihtoi nimensä ensin *Digital Unixiksi* ja sitten *Tru64*:ksi. Näiden lisäksi Alphalla voi ajaa *OpenVMS*:ää ja jotenkuten jopa Windows NT:tä.

DEC:n osti vuonna 1998 PC-raudasta tunnettu *Compaq*, jonka taas osti *Hewlett-Packard* vuonna 2002. Viimeiseksi Alphaksi jäi vuonna 2004 ilmestynyt 21364, kun HP päätti keskittyä superlaskennassa *PA-RISC*- ja *IA-64*-laitteistoihin.

International Business Machines

Suurista keskuskoneistaan ja PC:stä tunnettu tietotekniikkapioneeri IBM tuli Unix-työasemamarkkinoille vuonna 1986. *IBM 6150* eli *RT PC* perustui PC-koneissa käytettyyn *PS/2*-runkoon, mutta suorittimeksi oli vaihdettu IBM:n oma *ROMP*. ROMPia voidaan pitää ensimmäisenä RISC-mikroprosessorina, olihan IBM kehittänyt sen jo 1981. IBM oli kuitenkin pitänyt sitä pöytälaatikossa vuosikaudet, joten muut ehtivät tuotteistaa riscinsä ensin.

ROMPin syrjäytti vuonna 1990 *POWER*, jota käyttävät työasemat, palvelimet ja supertietokoneet saivat yhteisen nimen *RS/6000*. POWERin pikkuveljeksi tuli *Applen* ja *Motorolan* kanssa yhteistyössä kehitetty *PowerPC*, jota on käytetty myös tavallisten kuolevaisten koneissa.

IBM vaihtoi järjestelmäperheidensä brändäystä vuosituhannen vaihteessa, ja RS/6000:sta tuli tällöin *System p*. Myöhemmin System p yhdistettiin *System i:hin* eli entiseen *AS/400:aan*, jolloin muodostui *Power Systems* -niminen palvelin- ja supertietokoneiden mallisto. Nämä koneet tukevat IBM:n oman Unix-variantin *AIXin* lisäksi myös Linuxia ja AS/400:n peruja olevaa *IBM i* -käyttöjärjestelmää.

Suurkoneluokan mikroprosessorien kehitys on IBM:llä edelleen hyvin käynnissä. POWER-suorittimien ohella IBM on kehittänyt *zEC*-suorittimia, jotka huipputehokkuudestaan huolimatta ovat edelleen konekieliyhteensopivia IBM:n 60-luvun lippulaivan *S/360*:n kanssa.

Hewlett-Packard

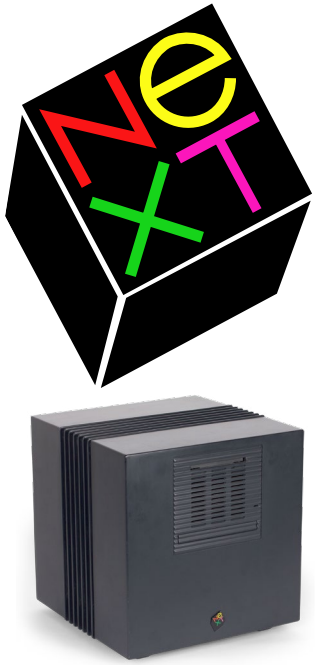
Hewlett-Packard on valmistanut erikokoisia tietokoneita ja laskimia oheislaitteineen 1960-luvulta alkaen. Unix-maailmaan HP siirtyi vuonna 1984 aloittamansa *HP 9000* -malliston myötä. Lisäksi HP osti vuonna 1989 *Apollo Computerin*, jolla oli omia työasemiaan.

Varhaiset HP 9000:t käyttivät 68000-suorittimien lisäksi myös pinopohjaisia *FOCUS*-suorittimia. Näistä siirryttiin 80-luvun loppupuolella HP:n omaan *PA-RISC*-arkkitehtuuriin. PA-RISCin tilalle tuli vuonna 2003 Intelin *IA-64* eli *Itanium*, minkä myötä fiktiiviseen HAL 9000:een viittaavasta mallinimestä vihdoinkin luovuttiin.

HP:n Unix-käyttöjärjestelmä *HP-UX* on monille edelleen kirosana. Vaikka se onkin sinänsä tehokas ja luotettava, se tekee monet asiat eri tavoin kuin muiden valmistajien unixit. HP:n tekstipäätteet eivät ole VT-yhteensopivia, oletusarvoinen C-kääntäjä soveltuu lähinnä vain HP:n oman koodin kääntämiseen, merkistöt ja komentoargumentit ovat erilaiset ja niin poispäin.

Suosituimpia PA-RISC-työasemia ovat olleet PC:n kilpailijoiksi 1990-luvun puolivälissä tarkoitetut 32-bittiset *712* ja *715*, joihin PC:n näppäimistöt, hiiret ja näytöt käyvät suoraan, mutta jotka päihittävät esimerkiksi 2D-näytönkäsittelyn nopeudessa aikakauden PC:t reilusti.

Nykyiset HP-UXia ajavat koneet ovat Itanium-pohjaisia *HP Integrity* -palvelinkoneita. HP-UXin lisäksi ne ajavat myös Linuxia, Windows Serveriä ja yritysostojen myötä HP:lle tulleita *NonStopia* ja *OpenVMS:ää*. Intel jatkaa Itanium-suoritinten kehitystyötä edelleen.



NeXT

Steve Jobsin vuonna 1985 perustaman NeXTin historia jäi varsin lyhyeksi, mutta se sai 1990-luvun kynnyksellä julkaistua muutaman Unix-työaseman, jotka ovat merkittäviä muutenkin kuin kuutionmuotoisten koteloidensa vuoksi.

NeXT Computer, *NeXTcube* ja *NeXTstation* perustuvat Motorolan 68030- ja 68040-suorittimiin ja multimediapotkua antavaan 56001-DSP:hen. Tarjolla oli myös erillinen RISC-kiihdytinkortti omine näytönohjaimineen, mutta ohjelmisto ei koskaan hyödyntänyt sitä kunnolla. Korppujen sijaan NeXTit käyttävät magneto-optisia levyjä, joiden kapasiteetti on 256 megatavua.

Käyttöjärjestelmänä NeXTeissä on *NeXTSTEP*, joka on Mach-mikroytimeen ja BSD:hen perustuva Unix – eivätkä muutkaan yhtäläisyydet myöhempään Applen *Mac OS X:ään* ole sattumaa. Esimerkiksi järjestelmäohjelmistot on kirjoitettu Objective-C:llä, ja X-ikkunoinnin sijaan on käytössä oma grafiikkaratkaisu omine käyttöliittymäinnovaatioineen. NeXTSTEPistä on olemassa myös X86-, SPARC- ja PA-RISC-koneilla toimiva versio *OpenStep*.

Speedrunien vimma

Ennätysjahtia ja huimaa viihdettä

Speedrunit ovat nousseet netti- ja striimiyhteisön suosioon viime vuosien aikana – mutta mitä ne oikeastaan edes ovat? Yksinkertaisimmillaan speedrunissa läpäistään videopeli mahdollisimman nopeasti. Tarkemmin tutkiessa asia monimutkaistuu. Joudumme esimerkiksi vastaamaan uuteen kysymykseen: mikä ylipäätään lasketaan läpipelaamiseksi?

Teksti: Eric Hartin Kuvat: Sakari Leppä, Eric Hartin, Mikko Heinonen

Täytyykö koko peli pelata läpi? Kelpaako se, jos näkee lopputekstit? Entä jos, entä jos...?

Kysymyksiä riittää yhtä paljon kuin katsojiakin. Ei ole lainkaan tavatonta, että speedrun-videon kommenteissa epäillään, onko tässä todella pelattu peli läpi vai kenties huijattu. Tämän vuoksi speedruneihin on luotu kategorioita, joita on useita erilaisia jokaiselle pelille.

Yleensä kategorioiden määrä kielii siitä, että peliä on pelattu ja pelataan edelleen paljon. Uusia rajoituksia syntyy haasteen etsimisestä ja pelin pitämisestä mielenkiintoisena sitä paljon pelaavalle.

Hämärää historiaa

Speedrunien historiaa on vaikea selvittää, sillä niitä on todennäköisesti ollut yhtä kauan kuin läpipelattavia pelejä ylipäätään. Ennen nykymuotoista internetiä niiden levittäminen on kuitenkin ollut kuitenkin vaikeaa, sillä suoritus on pitänyt nauhoittaa VHS-kasetille tai DVD:lle.

Joitain vanhempia esimerkkejä löytyy silti. Ensimmäiset varsinaista julkisuutta saaneet speedrunit olivat Quaken Quake Done Quick -demot. Quakessa pelinsä pystyi nauhoittamaan pieneksi datatiedostoksi, joka sisälsi ainoastaan pelaajan toiminnot. Näin suoritus saatiin muotoon, jossa sen pystyi lataamaan vanhan koulukunnan modeemillakin.

Viimeaikainen speedrun-innostus johtunee siitä, että Youtuben ja Twitchin aikakaudella kuka tahansa voi nauhoittaa itsensä haluamaansa peliä tahkoamassa. Näin varsinkin vanhojen Nintendo- ja Super Nintendo -pelien suoritukset ovat saavuttaneet yleisön, joka arvostaa taitavaa pelaamista.

any%, 100%

Karkein jako speedrunien välillä voidaan tehdä niin sanottujen any% ja 100% -kategorioiden välillä. Siinä missä any%-peluun aikana tavoitellaan vain lopputekstejä, 100%-juoksussa toteutetaan myös jokainen pelin sisäinen tavoite.

Speedrunien työkalut ovat yleensä vapaat. Pelaaja voi käynnistää pelin uudestaan niin monta kertaa kuin haluaa tai skipata pelistä niin paljon kuin haluaa, kunhan käyttää pelistä löytyviä tekniikoita. Jos pelin saa pelaamalla tilaan, jossa esimerkiksi jokin valikoista muuttuu heksaeditoriksi, on tätä sallittua käyttää.

Glitchaaminen, eli pelien saattaminen hetkellisesti häiriötilaan, on yleistä varsinkin koneavusteisissa speedruneissa. Häiriön voi aiheuttaa monilla tavoin riippuen siitä, miten ja mille alustalle peli on tehty. Tästä on hyötyä etenkin silloin, kun se mahdollistaa pelin nopeamman läpipelaamisen. Jos glitchaamalla pääsee pelin ensimmäisestä kentästä suoraan viimeiseen, ei tätä suinkaan jätetä hyödyntämättä.

Samalla kun uusia tekniikoita löytyy, luodaan myös uusia speedrun-kategorioita. Jos Pokemonin läpäisee 20 sekunnissa, luodaan uusia kategorioita, joihin panostaa – 20 sekunnin läpipeluu kun on mielenkiintoinen vain sen yhden kerran. Siitä on vaikea parantaa tavalla, joka tekisi läpäisyn mielekkääksi toisella tai kolmannella kerralla. Haastetta ja mielenkiintoa kasvatetaan tästä hakemalla nopeinta läpipeluuta, kun tietyllä tavalla glitchaaminen ei olekaan sallittua.

Oma kategoria itsessään on niin sanottu TAS, eli Tool-Assisted Speedrun. Siinä missä käytännön any% ja 100% on suunniteltu pelattavaksi siten, että ne voi suorittaa oikea ihminen oikealla raudalla, TAS poistaa ihmisen yhtälöstä. Yleensä TAS:eissa käytetään emulaattoria, joka mahdollistaa pelaa-

Mieleenpainuvia speedrun-ennätyksiä

Ninja Gaiden, 12 minuuttia

Ninja Gaidenin ennätykset ovat siitä mielenkiintoisia, että pelin vanhuudesta ja läpipeluiden verrattaisesta lyhyydestä huolimatta peliä ei glitchata tekemään outoja asioita, vaan läpipeluut ovat rehtejä ja pikaisia juoksuja paikasta A paikkaan B.

Mega Man X & X2, 2 tuntia 43 minuuttia

Koneavusteisissa läpipeluissa kahden tai useamman pelin pelaaminen samoilla syötteillä, eli pelaamalla kahta peliä yhtaikaa yhdellä ohjaimella, on jo vanha juttu. Sen tekeminen oikeasti pelaamalla on sen sijaan harvinaisempaa. Hyvä esimerkki siitä, mitä voi tehdä, kun pistää tarpeeksi aikaa lempipeleihinsä.

Super Mario World, 1 minuutti 41 sekuntia

Super Mario World on Super Nintendon ehkä dokumentoiduin peli. Ei olekaan ihme, että aikojen saatossa siitä on löydetty jos jonkinlaisia kikkoja. Alle kahdessa minuutissa suoritettu suoraan lopputeksteihin hyppäävä läpipeluu näyttää helpolta, mutta taustalla on pikkutarkkaa muistimanipulaatiota pelin aikana.

Mega Man, 12 minuuttia 23 sekuntia
Myös Mega Man on dokumentoitu tarkkaan ja myös siitä on löydetty vuosien varrella lukemattomia temppuja. Mega Manista tehdyn koneavusteisen läpipeluun pääasiallinen tarkoitus on esitellä Mega Man -moottorin erikoisuuksia pikemminkin kuin juosta peli läpi ennätysajassa.

Quake, 48 minuuttia

Ja mikäli pelkkien glitchien tiiraaminen alkaa väsyttää, voi aina seurata Quakea. Quake Done 100% Quickestissa pelataan peli alle 50 minuutissa läpi – ja samalla ammutaan jokainen vihollinen ja tutkitaan jokainen salahuone.

misen ruutu ruudulta, itse kirjoitettuja peliä pelaavia botteja tai kumpaakin yhtä aikaa. TAS:issa näytetään missä pelin rajat kulkevat ja millaisia ne ovat.

Vaikka TAS on luotu näyttämään pelin rajat, tavoitellaan siinäkin usein viihdyttävää esitysmuotoa. Monesti päätavoite ei olekaan nopein läpipeluu, vaan pikemminkin pelin bugien ja ominaisuuksien esittely. TAS:in suunnittelu saattaa kestää useita kuukausia ja vaatia jopa vuosien ennakkosuunnittelua. Monia koneavusteisia läpipeluita varten saatetaan avata peli ja tutkia, missä ohjelmakoodin muistipaikat kulkevat.

Tavoitteet vaihtelevat

Speedrunin tekoa lähestytäänkin usein monesta kulmasta. Toisaalta tarkoituksena on saada kelloon nopein aika millä keinolla hyvänsä, mutta yhtälailla saatetaan tavoitella viihdyttävää läpipeluuta. Joskus nämä tavoitteet ovat toistensa kanssa yhteensopimattomat. Jos peli läpäistään hyppäämällä ensimmäisestä ruudusta lattian läpi suoraan lopputeksteihin, jää viihdearvo olemattomaksi. Tämä on kuitenkin täysin kelvollinen ja luvallinen tapa pelata.

Speedruneilla, kuten monilla muillakaan yhteisöillä, ei ole monoliittistä tuomarointielintä. Alun perin kolikkopelisuorituksia kirjaava Twin Galaxies on speedrun-yhteisön mielestä kelvoton kirjaamaan speedrun-suorituksia: se haluaisi nimittäin kieltää läpipeluut, joissa peli glitchataan.

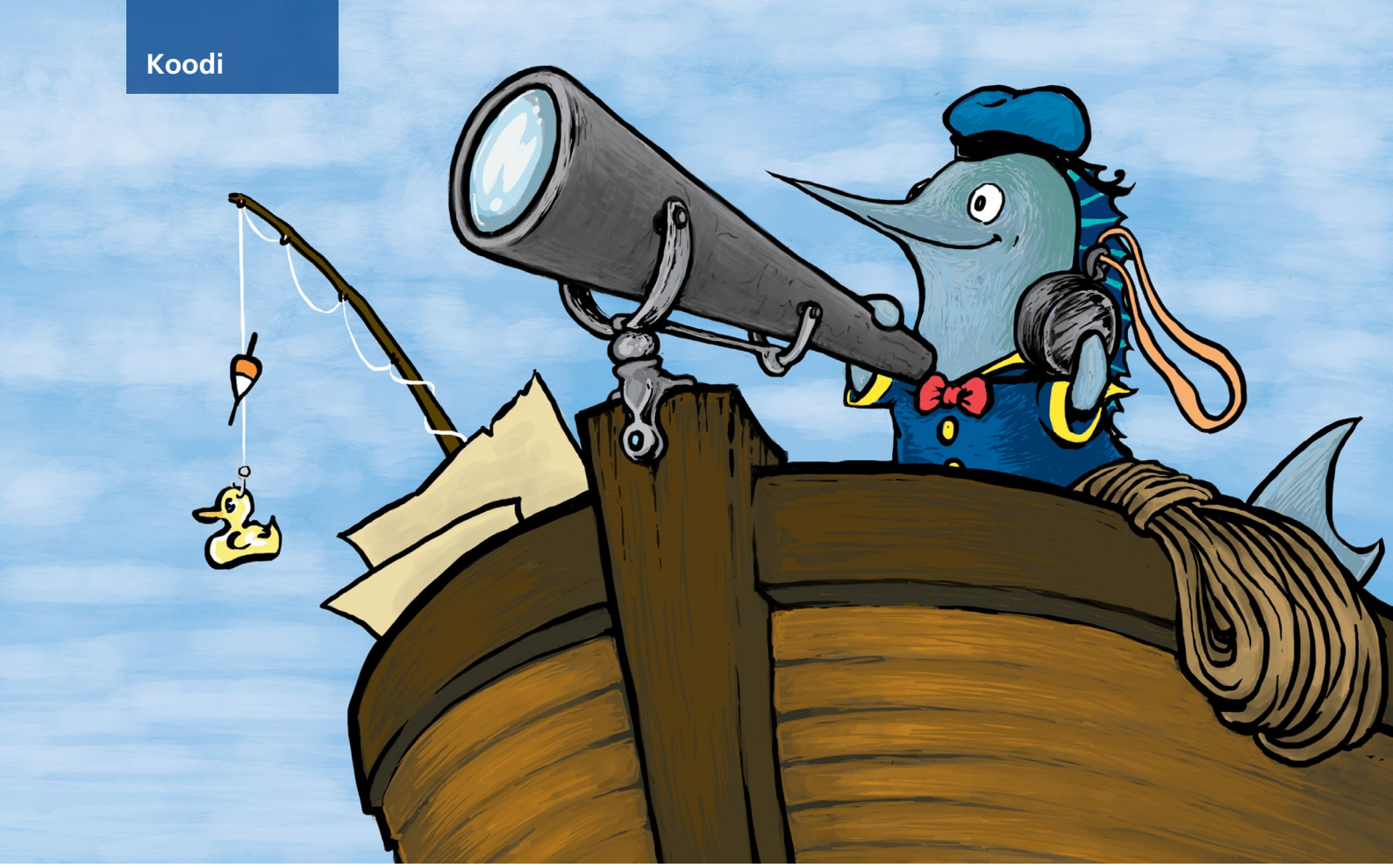
Kenties tällä hetkellä käytetyin sivusto on Speed Demos Archive, joka oli alkujaan omistautunut Quake-läpipeluille. Se myös hyväksyy läpipeluita useassa eri formaatissa: ne voidaan jaotella esimerkiksi osittaisiin juoksuihin ja juoksuihin, jotka on tehtävä yhdellä kertaa tai joissa sallitaan pelin käynnistäminen välissä. Sivusto hyväksyy pääosin vain alkuperäisillä alustoilla tehtyjä läpipeluita – emulaattorisuorituksia ei hyväksytä, sillä niistä on vaikeampi nähdä, onko peluu koneavusteinen vai ei.

Koneavusteisille läpipeluille onkin oma sivustonsa, TASvideos. Ihan kaikkea ei sekään hyväksy. Sivustolla listataan rutkasti asioita, joiden täytyy olla kunnossa, ennen kuin nauhoitus hyväksytään. Pääasiassa tämä tarkoittaa, että peliä ei ole muokattu millään tavalla, ja että emulointi täyttää tietyt hyvin tarkat kriteerit. Toisekseen läpipeluun tulee olla nopeampi kuin aikaisemmat suoritukset päästäkseen sivustolle – tai ainakin huomattavasti viihdyttävämpi.

Jos aihe kiinnostaa eivätkä speedrunit olleet ennestään tuttuja, tämän parempaa aikaa niiden löytämiselle ei olekaan. Eritoten kesällä nähtävä Summer Games Done Quick on uusienkin speedrun-katsojien mieleen, sillä sen aikana nähdään kymmenien pelien läpipeluut oivallisen selostuksen ja kikkojen selitysten kera.

Lisätietoja:

- Speed Demos Archive
  http://www.speeddemosarchive.com
- Summer Games Done Quick
  http://www.gamesdonequick.com
- TASvideos http://tasvideos.org



Mittausdataa irti Jollasta

Sailfish-ohjelmointi, osa 2

Skrollin numerossa 2014.2 tutustuttiin Sailfish-ohjelmoinnin perusteisiin. Nyt on sopiva aika ottaa peräsimestä tukevampi ote ja suunnata syvemmille vesille. Aloitetaan purjehdus opettelemalla Jollan antureiden käyttöä.

Teksti ja koodi: Asser Lähdemäki
Kuvat: Mitol Meerna, Asser Lähdemäki

Jolla-puhelin sisältää älypuhelimille tyypillisiä antureita ja oheislaitteita, esimerkiksi kompassin, kiihtyvyysanturin ja gyroskoopin. Tässä artikkelissa opettelemme hyödyntämään niitä ohjelmallisesti. Oletamme, että Sailfish Silica -käyttöliittymien oliot ovat lukijalle tuttuja. Keskitymme artikkelissa QML-rajapintoihin, mutta käytettäville QML-objekteille on lähes suorat vastineet myös Qt:n C++-rajapinnassa.

Artikkelin esimerkkiohjelma on testattu Qt 5.2- ja Sailfish OS 1.1.4.29 (Äijänpäivänjärvi) -versioilla. Joillakin muilla versioilla esimerkit eivät ehkä toimi ilman pieniä muutoksia, vaikka rajapinnat pyritäänkin pitämään yhteensopivina eri versioiden välillä. Valitettavasti esimerkkiohjelmaa ei voi täysin testata Sailfishin kehitysympäristön virtuaalikoneessa, sillä useimmille Jollan oheislaitteille ei toistaiseksi ole minkäänlaista emulaatiota. Ohjelman testaamiseen tarvitaan siis oikea Jolla-laite.

Anturirajapinta

Antureita luetaan melko yksinkertaisen oliorajapinnan kautta. Se otetaan käyttöön lisäämällä määrittelyn alkuun seuraava rivi:

```
import QtSensors 5.2
```

Antureiden QML-rajapinta ei ole välttämättä puhelimessa valmiina, joten ohjelmaprojektiin on lisättävä ajonaikainen riippuvuus kyseisiin rajapintoihin. Käytännössä se tarkoittaa seuraavan rivin lisäämistä projektin .yaml-tiedoston Requires-kohtaan:

```
qt5-qtdeclarative-import-sensors >= 5.2
```

Näin tarvittavat rajapinnat asennetaan puhelimeen samalla kuin itse sovelluskin.

Rajapinta perustuu Sensor-olioon, josta periytetään kaikille anturityypeille oliot. Tärkeimmät perityt ominaisuudet ovat dataRate, active ja reading. dataRate-ominaisuuden avulla annetaan mittaustaajuus hertseinä. Asettamalla active arvoon true anturi alkaa lähettää mittausarvoja.

Anturi	Perusolio	Lukemaolio	Lukemaolion ominaisuuksia
Kompassi	Compass	CompassReading	*azimuth*: Tyypiltään qreal. Kertoo kulman suhteessa pohjoiseen. *accuracy*: Kertoo lukeman tarkkuuden (arvo 0–1).
Kiihtyvyysanturi	Accelerometer	AccelerometerReading	*x, y, z*: Tyypiltään qreal. Ilmoittavat kiihtyvyyden akseleiden suuntaan.
Gyroskooppi	Gyroscope	GyroscopeReading	*x, y, z*: Tyypiltään qreal. Ilmoittavat kulmanopeuden akseleiden ympäri.

Taulukko 1. Anturit ja lukemaolioiden ominaisuuksia.

Arvot luetaan reading-ominaisuuden avulla. Se osoittaa lukemaolioon, joka periytyy Reading-oliosta. Kaikki lukemaoliot perivät ominaisuuden timestamp, jossa on viimeisimmän mittausarvon aikaleima mikrosekunteina. Muut ominaisuudet riippuvat anturista ja sen lukemaoliosta.

Antureiden aikaleimat riippuvat rajapinnan toteutuksesta, ja eri anturei den aikaleimojen ei taata olevan synkronoituja keskenään. Saman anturin aikaleimojen erotukset kertovat tarkasti kahden mittauksen välisen ajan, mutta esimerkiksi kiihtyvyys- ja kulmanopeuslukemien aikaleimojen erotus ei välttämättä anna mittaustapahtumien välistä aikaa luotettavasti.

Sensor-olion muista ominaisuuksista mainittakoon skipDuplicates ja axesOrientationMode. Ensin mainitun avulla estetään lukeman muutoksesta ilmoittava readingChanged-signaali, kun perättäiset lukemat ovat samat. Jälkimmäisellä asetetaan lukeman tulkintamoodi, kun laite käännetään eri asentoon. Oletuksena käytetään puhelimen pystyasentoa riippumatta laitteen todellisesta asennosta.

Kompassin käyttö

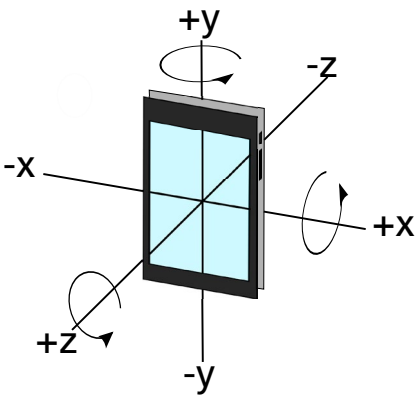
Tämän artikkelin esimerkkiohjelma on jaettu Sailfishille ominaisesti eri sivuihin. Jokaisen esitellyn anturin lukemiselle on omat sivunsa. Koodilistauksissa kolme pistettä tarkoittaa, että kohdasta on poistettu selitettävän asian kannalta epäolennaista koodia. Artikkelissa käytetyt anturioliot ovat taulukossa 1. Kolmannessa sarakkeessa on kyseisen anturityypin lukemaolio ja neljännessä sen hyödyllisimmät ominaisuudet.

Listauksessa 1 on kompassin käyttöesimerkki. Kompassianturi mittaa maan magneettikenttää ja antaa tulokset sena x-, y- ja z-komponentit. Akselit on oletuksena sidottu puhelimeen, kuten kuvassa 1 on esitetty. Arvot siis muuttuvat puhelimen asennon mukaan.

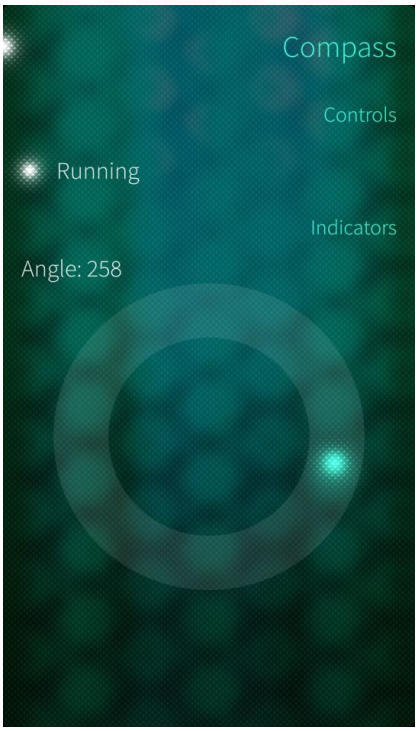
Kompassiolio ottaa anturilta saadut arvot ja laskee niistä maan magneettisen pohjoisnavan suuntaisen viivan ja positiivisen y-akselin suunnan välisen kulman. Toisin sanoen puhelimen USB-liittimen puoleinen pää vastaa perinteisen kompassin neulan pohjoiseen osoittavaa päätä. Olion laskeman arvon havainnointia varten on toteutettu CircleIndicator-komponentti, joka näkyy kuvassa 2.

```
import QtQuick 2.0
import Sailfish.Silica 1.0
import QtSensors 5.2

import "../components"

Page {
  id: root
  property bool readingReady: sensor.reading != null

  Compass {
    id: sensor
    active: activationSwitch.checked
    dataRate: 25
    property int angle: readingReady ? reading.azimuth : 0
    property real accuracy: readingReady ? reading.calibrationLevel : 0
  }

  Column {
    id: content
    width: parent.width
    PageHeader {
      title: qsTr("Title")
    }

    SectionHeader {
      text: qsTr("Controls")
    }

    TextSwitch {
      id: activationSwitch
      text: qsTr("Running")
      anchors.horizontalCenter: parent.horizontalCenter
      checked: true
    }

    ... // Muita ohjauskomponentteja

    SectionHeader {
      text: qsTr("Indicators")
    }

    CircleIndicator {
      anchors.horizontalCenter: parent.horizontalCenter
      anchors.top: content.bottom
      value: Math.abs(360 - sensor.angle)
      maxValue: 360
    }
  }
}
```

Listaus 1. Kompassisivun koodi.



Kuva 1. Akselit ja positiiviset kiertosuunnat.



Kuva 2. Esimerkkiohjelman kompassisivu.

Liiketilan ja asennon muutokset

Kiihtyvyyteen ja kulmanopeuteen päästään käsiksi Accelerometer- ja Gyroscope-olioiden kautta. Kuvassa 1 on vakioasetuksilla käytetyt akselit ja kiertosuunnat, joiden suhteen kiihtyvyys ja kulmanopeus mitataan.

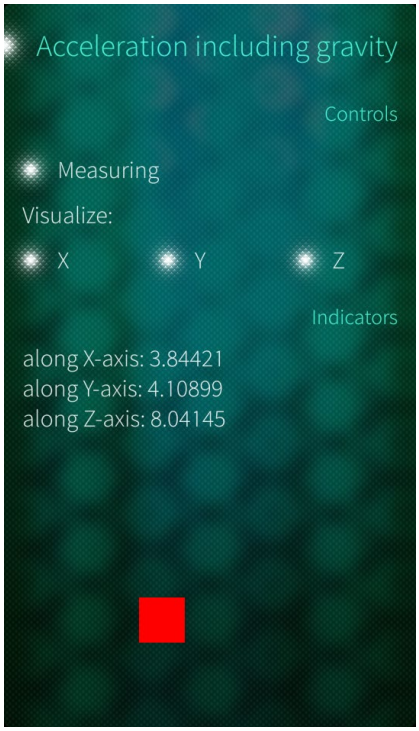
Kiihtyvyyden mittauksessa on otettava huomioon, että painovoiman aiheuttama putoamiskiihtyvyys on mukana mittaustuloksessa. Painovoiman vaikutuksen voi poistaa laskennallisesti, joskin tulos on epätarkka. Kiihtyvyys ilmoitetaan perusyksikössä m/s^2.

Kulmanopeuden mittaus kertoo, kuinka nopeasti laite pyörii akseleidensa ympäri. Esimerkiksi jos laitetta pyöräyttää pöydällä, mittaus antaa yhdelle akselille aluksi suuren arvon, joka sitten pienenee pyörimisen hidastuessa. Kulmanopeudet ilmoitetaan radiaaneina sekunnissa.

Listauksessa 2 on ohjelmakoodi kiihtyvyyden lukemiseen, ja ohjelman käyttöliittymä näkyy kuvassa 3. Ohjelma on monilta osin sama kuin kompassiesimerkissä, joten yhteiset osat on merkitty koodissa kolmella pisteellä.

Esimerkkiohjelmassa näkyvä punainen neliö liikkuu pysty- ja vaakasuunnassa kiihtyvyyden y- ja x-komponenttien mukaan. Z-komponenttia havainnollistetaan muuttamalla neliön kokoa. Ohjelman käyttöliittymän kytkimillä X, Y ja Z voidaan asettaa akseleiden mittauksen havainnollistus pois päältä.

Kulmanopeuden mittaussivu on lähes identtinen kiihtyvyyden mittaussivun kanssa, ja siinä käytetään samaa havainnollistustapaa. Ainoastaan mittaustulosten etumerkkejä on muutettu, jotta havainnollistuksessa neliö liikkuisi luonnollisiin suuntiin.



Kuva 3. Kiihtyvyysmittaussivu. Kiihtyvyydet on ilmoitettu lukuarvoina ja havainnollistettu liikkuvalla neliöllä.

```
...

Accelerometer {
  id: sensor
  alwaysOn: true
  active: activationSwitch.checked
  dataRate: 10
}

property bool readingReady: sensor.reading != null

Column {
  id: content
  width: parent.width
  PageHeader {
    title: qsTr("Acceleration including gravity")
  }

  SectionHeader {
    text: qsTr("Controls")
  }

  ...

  Row {
    width: parent.width
    TextSwitch {
      width: parent.width / 3
      id: xSwitch
      text: "X"
    }
    ... // Y ja Z TextSwitchit
  }

  SectionHeader {
    text: qsTr("Indicators")
  }

  Label {
    x: Theme.paddingLarge
    width: parent.width - 2 * Theme.paddingLarge
    text: qsTr("along X-axis: %1<br>along Y-axis: %2<br>"
             + "along Z-axis: %3")
    .arg(readingReady ? sensor.reading.x : 0)
    .arg(readingReady ? sensor.reading.y : 0)
    .arg(readingReady ? sensor.reading.z : 0)
  }
}

AccelerationIndicator {
  anchors.bottom: parent.bottom
  height: parent.height - content.height
  x_pos: xSwitch.checked ? (readingReady ? -sensor.reading.x : 0) : 0
  y_pos: ySwitch.checked ? (readingReady ? sensor.reading.y : 0) : 0
  z_pos: zSwitch.checked ? (readingReady ? sensor.reading.z : 0) : 0
  max_pos_abs: 15
}
}
```

Listaus 2. Kiihtyvyysanturin lukeminen.

Eikä se tähän lopu

Jolla tarjoaa monia muitakin antureita, joista osalle on toteutus esimerkkiohjelman koko versiossa. Esimerkkiohjelma on saatavissa sivulta http://skrolli.fi/2015.2. Muita antureita ovat esimerkiksi valoisuus-, läheisyys- ja näpäytysanturit.

Valoisuusanturille on kaksi oliota, joista toinen mittaa suoraan valoisuutta. Toinen valitsee anturin arvojen perusteella määrätyn arvon, esimerkiksi ”hämärää” tai ”kirkasta”. Läheisyysanturi kertoo, onko puhelimen yläosassa sijaitseva anturi peitetty vai ei. Näpäytysanturilla voi toteuttaa näpäytyseleitä.

Lisäksi esimerkkiohjelmassa on toteutettu muita sivuja, jotka havainnollistavat esimerkiksi paikannusta, järjestelmän rajapintoja, äänitehosteiden toistamista ja kameran käyttöä. Näitä ominaisuuksia käsitellään mahdollisesti Skrollin tulevissa numeroissa. Parannusehdotukset esimerkkisovellukseen ovat tervetulleita.

Lisätietoa
QML-anturirajapinnan dokumentaatio:
http://doc.qt.io/qt-5/qtsensors-qmlmodule.html

Croutonilla lisäkiiltoa Chromebookiin

Googlen selainpohjainen Chrome OS on lipunut vaivihkaa valtavirran käyttöjärjestelmäksi, erityisesti edullisten Chromebook-pikkukannettavien ansiosta. Chrome OS -laitteita voisi hyvin kutsua Google-päätteiksi, sillä ne on sidottu tiukasti hakukonejätin verkkopalveluihin aina sisäänkirjautumisesta lähtien. Rajoittunutta ohjelmatarjontaa ja riippuvuutta pilvipalveluista voi onneksi helpottaa asentamalla laitteeseen täysiverisen Linux-työpöydän, joka toimii rinnakkain Chrome OS:n kanssa.

Teksti: Markku Reunanen, Tero Heikkinen Kuvat: Markku Reunanen, Tero Heikkinen, Chris Helenius

Chrome OS:n sydämessä sijaitseva Linux-ydin ei ole samalla tavoin rajoittunut kuin esimerkiksi Androidin, vaikka konepellin alle pääsy onkin oletuksena tehokkaasti estetty. Googlella työskentelevän David Schneiderin luomus, Crouton, on näppärä keino asentaa laitteisiin tavallisia Linux-ohjelmia, joilla muuten puutteellinen ohjelmistovalikoima laajenee. Merkittävä hyöty perinteisempään dualboot-järjestelyyn verrattuna on se, että konetta ei tarvitse käynnistää uudelleen, vaan sekä natiiveja että Linux-ohjelmia voi ajaa samaan aikaan. Hankkeen sivut ja tiedostot sijaitsevat osoitteessa https://github.com/dnschneid/crouton.

Crouton ei ole virtuaalikone kuten vaikkapa VirtualBox tai VMware. Lyhyesti selitettynä kyse on chroot-vankilasta, jossa Linux-jakelun kuten Ubuntun kaikki tiedostot säilytetään yhdessä Chrome OS:n hakemistossa (/usr/local/chroots). Vankilasta ei ole pääsyä kuin hyvin rajallisesti Chromen puolelle, mikä pitää käyttöjärjestelmätiedostot turvallisesti erossa toisistaan – Croutonilla on siis käytännössä mahdotonta sekoittaa konetta käyttökelvottomaan kuntoon.

Koneen valinta

Tämän artikkelin pohjana ovat kokemukset kahdesta Acerin Chromebook-kannettavasta, C710:stä ja jonkin verran uudemmasta C720:stä. Molemmat ovat varmoja valintoja Crouton-koneeksi, sillä ne ovat laajalle levinneitä, jo iäkkäitä malleja – ja juuri siksi hyvin tuettuja. Croutonin voi toki asentaa muillekin Chromebookeille, mutta ARM-pohjaisissa laitteissa ei tueta esimerkiksi grafiikkakiihdytystä lainkaan, joten hyödyt jäävät niillä selvästi pienemmiksi. Tekniset tiedot on syytä lukea huolella ennen hankintaa tai asennusta, sillä Acerinkaan kaikki Chromebookit eivät ole enää Intel-pohjaisia.

Vaikka C720:ssä on nopeampi suoritin, se ei kuitenkaan ole itsestään selvästi parempi vaihtoehto, sillä muisti on vanhemmasta mallista poiketen integroitu kiinteästi emolevylle ja asemaksi kelpaa vain M.2/NGFF-tyyppinen SSD-kiintolevy. Kannattaakin ennen ostopäätöstä tarkistaa, onko koneessa kaksi vai neljä gigatavua muistia. Koneet eroavat muutenkin varustukseltaan: C710:ssä on vielä Ethernet sekä HDMI:n lisäksi VGA-liitin, siinä missä tuoreemmassa mallissa pitää pärjätä pelkällä langattomalla verkolla ja HDMI:llä.

C720:ssä on vakiona Croutonin kannalta onnettoman pieni, yleensä 16-gigainen SSD. Sen voi helposti vaihtaa, mutta laitekokonaisuuden hinta nousee samalla ja takuuhuollossa saatetaan nikotella rikkinäisen tarran vuoksi.

Kehittäjätilan käynnistysruutu – ei paineta sitä välilyöntiä!

Ennen aseman vaihtoa Chrome OS:n järjestelmätiedostot pitää tallentaa vähintään neljän gigatavun muistitikulle, josta järjestelmän voi palauttaa uudelle asemalle. Varmuuskopion tekeminen on muutenkin kannatettava ajatus, koska sillä saa aina pelastettua mahdollisesti sotketun asennuksen. Palautustikun luominen onnistuu Chrome Web Storesta löytyvällä Chromebook Recovery Utility -ohjelmalla.

Kehittäjätila ja Croutonin asennus

Jotta Croutonin asennus olisi mahdollista, pitää Chromebook ensin laittaa kehittäjätilaan. Vaihto tapahtuu näissä malleissa järjestelmän palautusnäkymästä, johon pääsee painamalla Escape- ja virkistä/F3-näppäimet pohjaan ja näpäyttämällä sitten virtanappia. Ruutuun tulee iso keltainen huutomerkki ja varoitus Chrome OS:n puuttumisesta tai vahingoittumisesta. Tästä ei kuitenkaan tarvitse olla huolissaan. Ctrl-D vie jälleen toisen näköiseen ruutuun, jossa varmistetaan siirtyminen kehittäjätilaan. Ennen pitkää kone käynnistyy kokonaan uudestaan ja pyytää tavalliset sisäänkirjautumis- ja asetustiedot.

Jatkossa kone siirtyy aina käynnistyessään valkoiseen ruutuun, jossa on tietokoneen kuva ja punainen huutomerkki. Alla on kehotus painaa välilyöntiä, mitä ei kuitenkaan kannata tehdä, ellei halua poistaa kehittäjätilaa ja samalla koko Crouton-asennusta. Chrome OS lähtee käyntiin painamalla Ctrl-D tai odottamalla hetken, minkä jälkeen siirrytään tavalliseen sisäänkirjautumiseen.

Tässä vaiheessa käydään lataamassa Croutonin sivulta asennusohjelma, joka tallentuu muiden tiedostojen tapaan Downloads-hakemistoon. Painamalla selaimessa Ctrl-Alt-T aukeaa crosh-komentotulkki (Chrome Shell). Kone saattaa kysyä käyttäjätunnusta, joka on vakiona ”`chronos`”. Salasanaa ei tarvita. Komennolla ”`shell`” päästään vuorostaan Linuxista tuttuun komentorivikehotteeseen. Sitä kautta voidaankin jo asentaa itse chroot-vankila:

```
sudo sh ~/Downloads/crouton -t lxde
```

Tässä tapauksessa koneeseen menisi Ubuntun oletusversio (12.04) sekä kallisarvoista muistia säästävä kevyt LXDE-työpöytäympäristö. Muita vaihtoehtoja ovat esimerkiksi Xfce tai Unity. Kun Crouton on saanut asennuksen loppuun, voidaan käynnistää haluttu chroot-vankila, esim. ”`sudo startlxde`”, minkä jälkeen LXDE:n pitäisi ilmestyä näkyviin lähes välittömästi.

Perustoiminta ja tärkeimmät säädöt

Kun työpöytä on käynnissä, voi ympäristöjen välillä siirtyä näppäinyhdistelmillä Ctrl-Alt-Shift-paluu/F1 ja -eteenpäin/F2. Joissakin versioissa shiftin painaminen ei ole tarpeen. Jos LXDE:n ruutu jää mustaksi, voi näytön virkistää painamalla Ctrl-Alt-virkistä/F3. Uusimmissa versioissa tämä tosin tuntuu olevan jo tarpeetonta. Vaikka käynnistyskomennot saattavat näyttää alkuun sekavilta, päästään työpöydälle itse asiassa nopeammin kuin monissa pöytäkoneissa tai Chromebookille kehitetyssä dualboot-Linuxissa.

Chrome OS:n ja chroot-vankilan välillä ei ole lähtökohtaisesti juurikaan yhteyksiä. Eri ympäristöt jakavat sentään saman Downloads-kansion, mikä onkin moniin tarkoituksiin aivan riittävää. Vielä tiiviimpää yhteyttä eri työpöytien välillä halajavan kannattaa hakea Web Storesta Crouton Integration -laajennus, jonka kautta leikepöydän saa jaettua ympäristöjen välillä. Croutonin kautta täytyy lisäksi asentaa hieman hämmentävästi nimetty *extension*-laajennus. Xiwi taas tuo mukanaan mahdollisuuden ajaa Linux-työpöytää Chrome-ikkunan sisällä, mikä on kylläkin harmillisen hidasta.

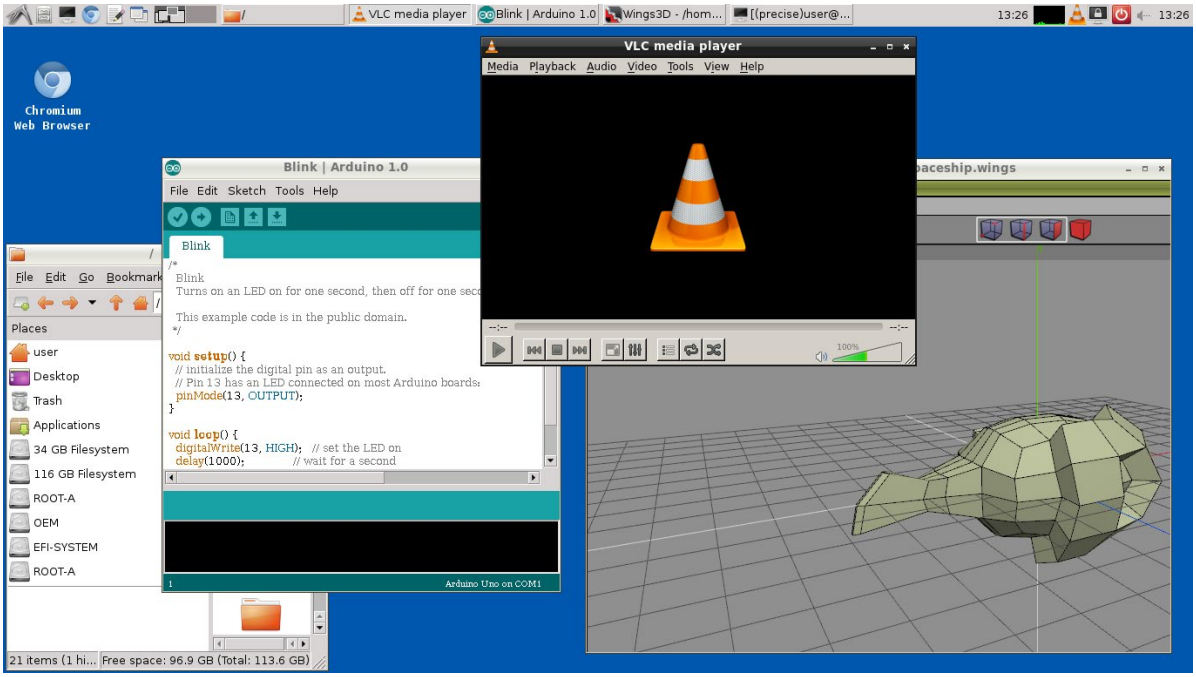
Alkuinnostuksen jälkeen huomataan pian, etteivät kaikki yksityiskohdat olekaan vielä kunnossa. Skandinaaviset näppäimet, äänet ja erinäiset laiteajurit saattavat tuottaa aluksi pieniä ongelmia. Osassa Linux-työpöytäympäristöistä on valmiit asetusohjelmat näppäimistölle; minimalistisessa LXDE:ssä ääkköset ja erikoismerkit saa ensi hätään käyttöönsä komentamalla ”`setxkbmap fi`”. Jos vaikutukset jäävät puolittaisiksi, voi syynä olla tyhjäksi jäänyt /etc/default/locale-tiedosto, johon voi lisätä rivin ”`LANG=en_US.utf8`” ja käynnistää chroot-vankilan uudestaan.

Chromebookin näppäimet esimerkiksi näytön kirkkaudelle ja äänenvoimakkuudelle eivät nekään toimi suoraan, sillä ylärivin napit näyttäytyvät Linux-työpöydälle tavallisina funktionäppäiminä. LXDE:ssä oikoteitä voi lisätä tiedostoon .config/openbox/lxde-rc-xml. Hämärämpiä ongelmia voivat aiheuttaa laitetiedostojen väärät käyttöoikeudet. MIDI-laitteet alkoivat toimia esimerkkiasennuksessa vasta muuttamalla oikeudet seuraavalla käskyllä:

```
sudo setfacl -m u:omakäyttäjätunnus:rw / \
dev/snd/*
```

Ohjelmia, ohjelmia

Seuraavaksi on aika nauttia tehdyn työn digitaalisista hedelmistä. Aito Ubuntu-jakelu sisältää paljon sellaisia sovelluksia, joita ei ainakaan tois-



LXDE-työpöytä ja sekalaisia ohjelmia.

taiseksi ole saatavilla Chrome OS:lle. Ensimmäisiä ilmeisiä käyttökohteita on ohjelmointi: kääntäjiä, editoreita, versionhallintaohjelmia ja kirjastoja löytyy joka lähtöön. Esimerkkeinä toimiviksi todetuista kehitysympäristöistä mainittakoon tässä Processing ja Arduino. Chromebook tai Chromebox taipuu pienellä vaivalla myös pieneksi mukana kulkevaksi palvelimeksi.

Erilaiset pikaviestimet XChatista Skypeen ovat nekin helposti saatavilla. Chrome OS:llekin on kyllä esim. IRC-asiakkaita, mutta niiden laatu ei vakuuttanut, ja Skype puuttuu tällä hetkellä kokonaan. Muut tavanomaiset Linux-työkalut, kuten Gimp-kuvankäsittely, LibreOffice ja Audacity-ääni-editori, toimivat kuten pitääkin. Jopa 3D-mallinnusohjelma Blenderiä pystyy käyttämään kohtuullisesti, kunhan pitää odotuksensa realistisina. Videoiden katselu onnistuu sujuvasti monelta muultakin alustalta tutulla VLC:llä, joka toistaa eksoottisetkin tiedostot (jos ääni katkeilee, kannattaa vaihtaa PulseAudio käyttöön asetuksista).

Crouton on myös pelaajan ystävä. Jo pelkästään Ubuntun valikoimista löytyy jonkin verran viihdykettä ja emulaattorien kautta sitäkin enemmän. Testasimme pikaisesti VICEn (8-bittiset Commodoret), Fusen (ZX Spectrum), DOSBoxin (PC), UAE:n (Amiga), Hatarin (Atari ST), openMSX:n sekä ScummVM:n, jotka kaikki pyörähtivät Acereilla käyntiin moitteettomasti. Ainoaksi kauneusvirheeksi jäi VICEn heikosti toimiva koko ruudun tila, joka tosin lienee LXDE:n käyttämän Openboxin eikä Croutonin ongelma. Jopa Steam on mahdollista saada asennettua pienellä vaivalla, ja etenkin grafiikkaominaisuuksiltaan jykevämpi C720 jaksaa pyörittää kohtuudella vanhempia 3D-pelejä.

Croutonin rajoitukset

Asiat toimivat yleisesti ottaen hyvin, mutta aivan kaikki ei näin kokeellisessa ympäristössä tietenkään suju ilman kompurointia. Edes Chrome OS ei ole pomminvarma, ja kun sen kylkeen lisätään vielä Crouton, ilmenee toisinaan outoilua, jonka syy saattaa jäädä epäselväksi. Eräs toistaiseksi ratkaisematon pulma on se, että aikavyöhykkeet ja tiedostojen aikaleimat eivät noudata normaaleja käytäntöjä: usb-asemille tallentuvat tiedostot näkyvät Chromessa oikein, mutta muualla niiden päiväys on kaksi tuntia pielessä.

Chroot-vankila käyttää samaa ydintä kuin Chrome OS, mistä aiheutuu joitakin rakentavaa laatua olevia ongelmia. Ytimeen ei ole vakiona käännetty järin laajaa laitetukea, joten kovin erikoiset lisälaitteet eivät siten toimi. Sama koskee VirtualBoxin vaatimia moduuleja, joita ei voi kääntää normaalilla tavalla – hyvästi siis virtualisointi. Tavalliset HID-standardin mukaiset hiiret, näppäimistöt, joystickit ynnä muut ovat onneksi tuettujen joukossa. Harrastuneempi säätäjä voi toki kääntää ytimen itsekin, mutta moinen operaatio ei ole aivan Pertti Peruskäyttäjän ulottuvilla ja saattaa vaarantaa järjestelmän sujuvan toiminnan.

Muita vastaan tulleita pikku ongelmia ilmeni esimerkiksi usb-asemien liittämisen kanssa. Chrome OS valittaa joka kerran, kun LXDE:ssä lisää ja poistaa laitteen; tätä tosin tapahtui jostain syystä ainoastaan C720:llä. Chrome OS:n ohjelmistopäivityksistä saattaa niistäkin aiheutua päänsärkyä, jota helpottaa Croutonin päivitys vivulla -u. Ei siis kannata järkyttyä, jos jonakin päivänä Linux-työpöytä ei ilmestykään näkyviin.

Lopuksi

Kuten edeltä lienee jo käynyt ilmi, Croutonin kanssa askartelu ei sovi aivan kaikille. Pelkkään päivittäiseen nettiselailuun tai Google Docsin näppäilyyn Chrome OS on varsin riittävä ja sujuvampi kuin vaikkapa LXDE. Hiemankin Linux-taitoinen saa kuitenkin varsin kohtuullisella säätämisellä koneestaan irti monenlaista uutta, joten vaivannäkö kannattaa. Croutonin kautta asentuvat ohjelmointityökalut, grafiikka- ja musiikkiohjelmat, emulaattorit ja komentorivityökalut laajentavat Chromebookin käyttömahdollisuuksia kauas nettiselailun tuolle puolen.

Verkkovakoilun tekninen ulottuvuus

Tiedustelupalveluiden perinteinen verkkotiedustelu on laajentunut massavalvonnaksi: Digitaalisessa panopticonissa on elettävä olettaen, että näkymätön tarkkailija saattaa koska tahansa tutkia viestinnän ja päätelaitteen sisällön. Järjestelmän kerrotaan seulovan teroristeja, mutta viimeaikaisista paljastuksista ilmenee, kuinka kohteita ovat paitsi toisinajattelijat ja journalistit, myös tavalliset kansalaiset. Miten verkkovakoilu tapahtuu teknisesti? Entä voiko siltä voi suojautua?

Teksti: Markus Ottela
Kuvat: Mitol Meerna, Markus Ottela, Wikimedia / Matt Crypto, Electronic Frontier Foundation

Internetin kaltainen neutraali, sensuuriton, anonyymi ja yksityinen viestintäympäristö on välttämätön modernin tietoyhteiskunnan terveelle kehittymiselle. Monet autoritaaristen maiden hallinnot pelkäävät arabikevään kaltaista, sisäisistä jännitteistä syntyvää vallankumousta. Ne ovat siksi alkaneet vainota katalyyttina toimivia toisinajattelijoita, journalisteja sekä aktivisteja.

Samalla verkon sananvapautta rajoitetaan sensuroinnin ja massavalvonnan keinoin, ja perusteena käytetään niin sanotusti neljää infokalyptista hevosmiestä: terrorismi, huumekauppa, järjestäytynyt rikollisuus sekä pedofilia herättävät hyvin vahvoja tunteita. Ne salpaavat rationaalista ajattelua ja luovat sosiaalista painetta olla kyseenalaistamatta, ovatko hevosmiesten kiinniottoon käytetyt menetelmät tehokkaita, relevantteja, tai edes varsinainen päämäärä.

Yhdysvalloissa lukuisat ilmiantajat ovat tuoneet julkisuuteen epäkohtia siitä, kuinka teknologiaa väärinkäytetään henkilökohtaisen tai kansallisen edun tavoitteluun riippumatta siitä, miten valvonnasta on säädetty laissa tai ihmisoikeussopimuksissa. Epäeettinen verkkovalvonta on rakennettu kolmen pilarin varaan: juridiikan, politiikan sekä tekniikan. Ongelman ratkaisu edellyttää muutoksia jokaisessa niistä.

Tämä artikkeli tarkastelee ensisijaisesti massavalvonnan teknistä puolta sekä teknisiä suojautumismenetelmiä. Jotta verkkovalvonnasta voi saada yleiskuvan, on tärkeää ymmärtää sen historiaa ja kuinka tekniikka on ke-

Royal Air Forcen Menwith Hillillä sijaitsevat Echelon-järjestelmän radome-kuvut. Kuva: Matt Crypto.

hittynyt ja skaalautunut vanhentuneen lainsäädännön mahdollistamana.

Isoveli lapsenkengissä

Toisen maailmasodan päätyttyä Iso-Britannian ja Yhdysvaltojen väliaikainen tiedusteluyhteistyö jatkui 1946 allekirjoitetulla UKUSA-sopimuksella. Yksi ensimmäisistä elektronista viestintää vakoilevista ohjelmista oli Yhdysvaltojen National Security Agencyn (NSA) vuosina 1967-73 ajama Minaret, joka nauhoitti yli seitsemän tuhannen tarkkailulistalle valitun kohteen puheluita. 1960-luvulla tapahtui teknologinen harppaus, ja mannertenvälinen kommunikaatio siirtyi vähitellen satelliitteihin.

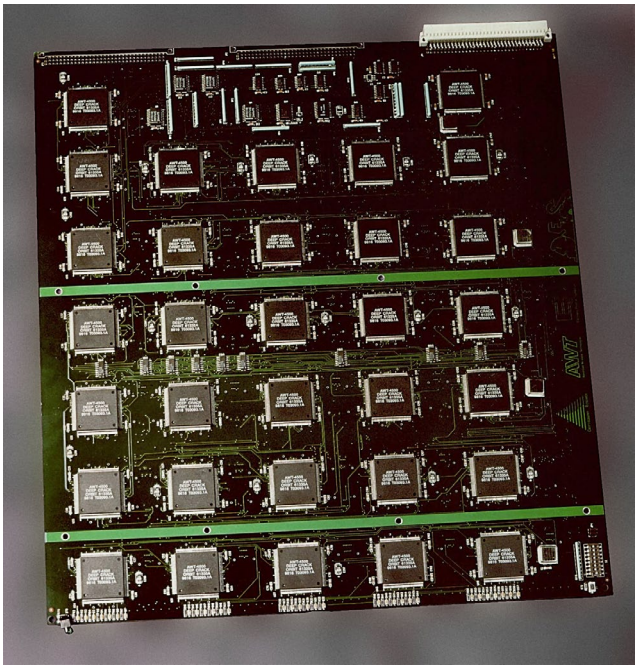
Tiedustelupalvelut seurasivat perässä, ja Iso-Britannian Government Communications Head Quartersin (GCHQ) ensimmäinen satelliittilinkkien kuunteluasema nousi Etelä-Britannian Morwenstowiin jo vuonna 1971. GCHQ ja NSA tekivät tiivistä yhteistyötä, ja seuraavan vuosikymmenen aikana valtioiden kuunteluasemat yhdistettiin Echeloniksi kutsutulla laajaverkolla. Liikenteen kuuntelu oli helppoa, sillä satelliittiliikenne oli salaamatonta, kuten myös 70-luvulla käyttöön tulleet valokuidut, ensimmäiset matkapuhelinverkot sekä Arpanet, josta kehittyi 1990-luvun alkuun mennessä kaupallinen Internet.

Echelonin satelliittitiedustelu vanhentui 2000-luvulla, mutta signaalitiedustelu pysyi ajan tasalla, kun uudempi tiedonsiirtoteknologia, kuten valokuitu sekä mikroaaltolinkit, kytkettiin järjestelmään. Merikaapeleiden kuuntelu onnistui vähitellen ympäri maailmaa, sillä 80-luvun puoliväliin mennessä myös muut englanninkieliset länsimaat – Kanada, Australia sekä Uusi-Seelanti – olivat liittyneet sopimukseen, muodostaen Five Eyes (FVEY) -nimisen yhteenliittymän.

Echelon paljastui vuonna 1988, kun Lockheed Martinin työntekijä Margaret Newsham vihelsi pilliin kertomalla lehdistölle, että Iso-Britannian Menwith Hilliltä käsin oli kuunneltu yhdysvaltalaisen senaattorin Storm Thurmondin puheluita. Uutinen herätti pelkoa vuosien 1972–74 Watergate-skandaalin toistumisesta, jossa opposition jäseniä salakuunneltiin. Tilannetta pahensi tiedustelua valvovan komitean haltuunsa saamat piirustukset, joista kävi ilmi, että poliittisten tahojen valvonta on alusta alkaen rakennettu osaksi Echelon-järjestelmää.

Euroopan Parlamentin valiokunta teetätti Echelonista selvityksen, jonka alustavasta versiosta Helsingin Sanomat uutisoi 27.2.1998 otsikolla ”Isoveli elää verkossa”. Uutisen mukaan ”NSA etsii Echelonin avulla tietoa potentiaalisista terroristeista, mutta suuri osa kerättävistä tiedoista koskee taloutta.”

EP:n 2001 julkaisema kertomus suositti juridisten ja poliittisten keinojen lisäksi kansalaisia ja julkishallintoa muun muassa salaamaan kommunikaationsa sekä suosimaan ja tukemaan avoimen lähdekoodin ohjelmistoja.



EFF:n Deep Crack -supertietokone.
Kuva: Electronic Frontier Foundation.

Kryptosota

Euroopan Parlamentin mahdollisuus suosittaa viestien vahvaa salaamista edellytti historiallisesti merkittävää poliittista prosessia, jonka lähtöasetelma on syynä järjestelmien haavoittuvuutteen vielä tänäkin päivänä.

Yhdysvalloissa käytiin 1990-luvulla niin sanottu kryptosota (crypto wars), jossa osapuolina olivat turvallista ja yksityistä viestintää vaativa cypherpunk-liike sekä kryptografian vientirajoituksia vaativa tiedusteluyhteisö. Vientirajoitusten vuoksi esimerkiksi Netscape joutui heikentämään selaimensa vientiversiossa RSA-salausalgoritmin avainpituuden 1024:stä 512 bittiin. Myös GSM:n A5/1- ja satelliittiyhteyksien A5-GMR-1-algoritmeille oli vientirajoitettuja maita varten omat heikennetyt versiot A5/2 ja A5-GMR-2.

Kryptografian vientirajoitusten kontrollointi kävi kuitenkin mahdottomaksi, sillä vahvat salausohjelmistot, kuten PGP, vuotivat ulkomaille sananvapauden turvaamana, koneluettavana paperivuorena. Vientirajoitukset haittasivat myös elektronisia kauppa- ja pankkiyhteyksiä. Osaltaan kaupallisen sektorin painostuksesta presidentti Bill Clinton poisti kaupallisen kryptografian vientä rajoittavalta ’munitions list’ -listalta vuonna 1996.

RSA-avainpituudet vapautuivat, ja vuotta myöhemmin Electronic Frontier Foundationin (EFF) Deep Crack -supertietokone mursi silloisen Data Encryption Standard (DES) -salauksen. Syntyi tarve uudelle symmetriselle salausalgoritmille. Nelivuotisen avoimen kilpailun jälkeen marraskuussa 2002 National Institute of Standards and Technology valitsi Rijndael-nimisen algoritmin uudeksi Advanced Encryption Standardiksi (AES).

Vahva kryptografia oli lopulta kaikkien käytettävissä ja Foundation for Information Policy Research julisti vuonna 2005 kryptosodan voitetuksi. Salausalgoritmien käyttöönotto tapahtui kuitenkin takkuillen, ja monesti ne eivät olleet oletuksena päällä.

Verkkovalvonta ei kuitenkaan päättynyt. Vuonna 2006 teleoperaattoriyhtiö AT&T:n teknikko Mark Klein paljasti yrityksen tiloissa sijaitsevan 641a-nimisen huoneen, jossa mannertenvälinen valokuituliikenne haaroitettiin NSA:n tietokoneille analysoitavaksi. Samana vuonna NSA:n vanhempi työntekijä Thomas Drake paljasti tietoja viestintää analysoivasta Trailblazer-ohjelmasta.

Kolmas paljastus tapahtui vuonna 2008, kun Thomas Tamm vuoti tietoja vuosina 2002–2004 toimineesta Stellar Wind -ohjelmasta, jolla kerättiin kansalaisten viestiliikennettä sekä tietoja näiden rahaliikenteestä ja verkkokäyt-

NSA:n Utahin osavaltion Bluffdaleen rakentama datakeskus.
Kuva: Electronic Frontier Foundation.

Upstream ja Prism ovat SSO:n kaksi tärkeintä ohjelmaa.

täytymisestä.

Näitä paljastuksia seurasi uutishiljaisuus, jonka rikkoivat lähinnä Wikileaksin vuonna 2011 julkaisemat valvontajärjestelmiä myyvien yritysten katalogit sekä Wired-lehden maaliskuun 2013 artikkeli Utahiin rakennettavasta NSA:n Mission Data Repository -datakeskuksesta.

Ajanlasku Snowdenin jälkeen

”Julkisesti väitämme, että havainnointikykymme hämärtyy, mutta todellisuudessa pääsymme parantuu jatkuvasti. Totuus on, että NSA ei koko historiansa aikana ole kerännyt enemmän kuin tällä hetkellä.”

-Edward Snowden, Citizenfour11

Toukokuussa 2013 Booz Allen Hamiltonilla työskentelevä NSA:n infrastruktuurianalyytikko Edward Snowden lensi Hong Kongiin. Siellä hän tutki dokumentintekijä Laura Poitraksen sekä The Guardianin toimittaja Glenn Greenwaldin kanssa tuhansia huippusalaiseksi luokiteltuja asiakirjoja, joita Snowden oli aiemmin lähettänyt Poitrakselle.

Uutisointi asiakirjoista alkoi jo samalla viikolla, ja aiheeseen liittyvät otsikot ovat täyttäneet lehtien etusivuja ympäri maailmaa samaa tahtia, kun asiakirjoja on osattu tulkita ja nimiä sensuroida.

Vaikka asiakirjat ovatkin tarjonneet tähän asti perusteellisinta tietoa valtioiden verkkovakoilumenetelmistä, on huomautettava, että vasta noin 4 000 sivua arvioidusta 57 000:sta on julkaistu. Snowdenin asiakirjat eivät valitettavasti pysty tarjoamaan täydellistä kokonaiskuvaa valvonnasta, sillä NSA luokittelee muun muassa Core Secrets -informaation ’Exceptionally Controlled Information’ -kategoriaan (ECI), jota ei kirjoiteta lainkaan paperille.

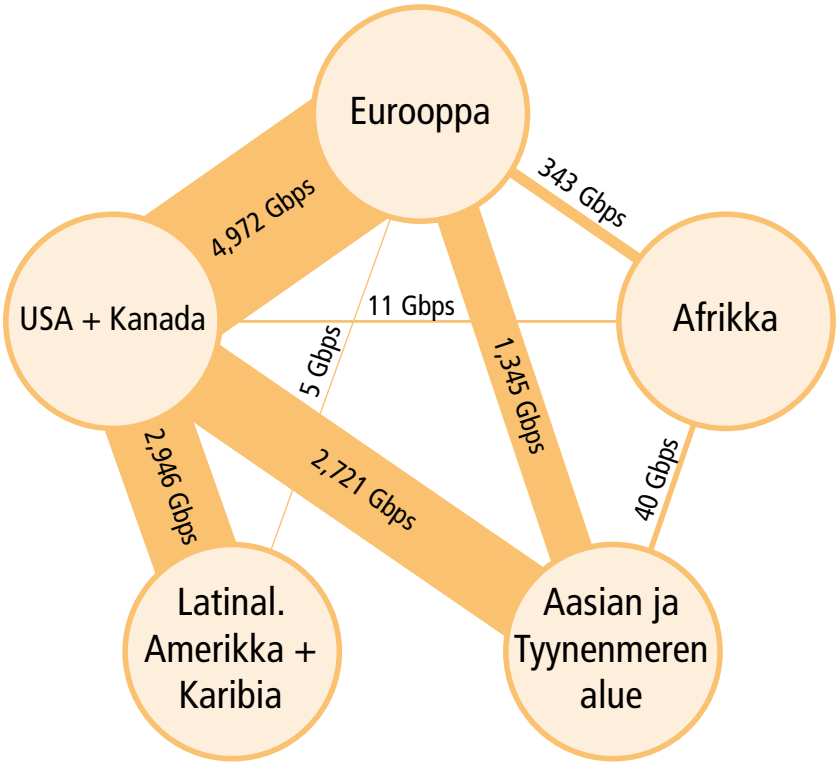
FVEY-jäsenmaat poislukien suurvaltojen verkkovakoilukyvyistä on saatavilla hyvin rajallisesti tietoa. Spekuloinnin sijaan todettakoon, että eri suurvaltojen rajojen sisällä tekniset keinot ovat korkeintaan NSA:ta vastaavalla tasolla. Suurvaltojen kansalaisten oikeus yksityisyyteen omalta hallinnolta on kuitenkin monesti huonommalla tolalla kuin FVEY:n kansalaisilla omissa jäsenmaissaan. Verkkohyökkäysten osalta suurvallat ovat samalla viivalla, sillä kaikkien osapuolten internetiin kytketyt laitteet tulevat samoista tehtaista. Lisäksi kaikki osapuolet kehittävät ja ostavat järjestelmiin kilpaa hyökkäysohjelmistoja.

FVEY-yhteenliittymän kyky harjoittaa globaalia massavalvontaa on kuitenkin poikkeuksellinen, koska suurin osa internetin runkoverkoista kulkee sen jäsenvaltioiden läpi. Lisäksi ylivoimainen osa ohjelmistoyrityksistä, internetin palveluista sekä internetin globaalin luottamuksen takaavista yrityksistä sijaitsee Yhdysvalloissa.

Snowdenin asiakirjoista käy ilmi, kuinka lakkautetut ohjelmat ovat jatkuneet uusien koodinimien alla hyväksikäyttäen jäsenvaltioiden etuoikeutettua roolia globaalissa verkossa. NSA:n Special Source Operations (SSO) on ottanut hoteisiinsa Stellar Windin ja laajentanut projektia niin, että pelkästään massavalvontajärjestelmän yläkategoriat tarvitsevat kahdeksan erillistä koodinimeä. Trailblazerin korvanneessa Turbulence-ohjelmassa näitä tarvitaan yhdeksän.

Verkkovakoilu

SSO:n Prism-järjestelmä mahdollistaa tietojen hakemisen ainakin Microsoftin, Yahoon, Googlen, Facebookin, PalTalkin, AOLin ja Applen palvelimilta. Ohjelman yhteistyöyritykset vastasivat Snowden-asiakirjojen väitteisiin vältellen, lähinnä kieltäen yk-



Internetin alueellinen kaistanleveyskapasiteetti maailmassa vuonna 2011.

Broad Oak seuraa TLS:n trendejä.

2. In recent years there has been an aggressive effort, lead by NSA, to make major improvements in defeating network security and privacy involving multiple sources and methods, all of which are extremely sensitive and fragile. These include: Computer Network Exploitation (CNE); collaboration with other Intelligence Agencies; investment in high-performance computers; and development of advanced mathematical techniques. Several ECI compartments may apply to the specific sources, methods, and techniques involved.

sittäisiä pääsymenetelmiä kuten takaportin (Google) tai suoran pääsyn (Facebook). Yksikään yrityksistä ei kuitenkaan kieltänyt suoraan yhteistyötä NSA:n kanssa.

Asiakirjoista käy myös ilmi, että ainoastaan liittovaltion poliisi FBI on yhteydessä Prism-ohjelman kumppaneihin. Tilaa sanavalinnoilla kikkailulle siis on, mutta ainoat keinot toteuttaa asiakirjojen väittämä pääsy sekä reaaliaikaiseen kommunikaatioon että palvelimelle tallennettuihin tietoihin on joko automatisoitu kanava palvelimeen tai jatkuva ”mies-välissä-hyökkäys” (man-in-the-middle attack) jokaista käyttäjää vastaan.

Toinen SSO:n ohjelma on Upstream, jonka alla toimii neljä ohjelmaa: Fairview ja Blarney kahdentavat Yhdysvaltojen sisäistä valokuituliikennettä yhteistyössä sikäläisten teleyritysten kanssa. Stormbrew vastaa Yhdysvaltojen-, ja Oakstar ulkomaiden rannikoilta nousevista merikaapeleista. Upstreamiin liittyy olennaisesti myös NSA:n USS Jimmy Carter -sukellusvene, jonka The Atlantic uutisoi vuonna 2006 kykenevän kuuntelemaan merenpohjassa kulkevia valokaapeleita. Lisäksi GCHQ:lla on Tempora-niminen, Stormbrew'tä vastaava runkoverkon kuunteluohjelma.

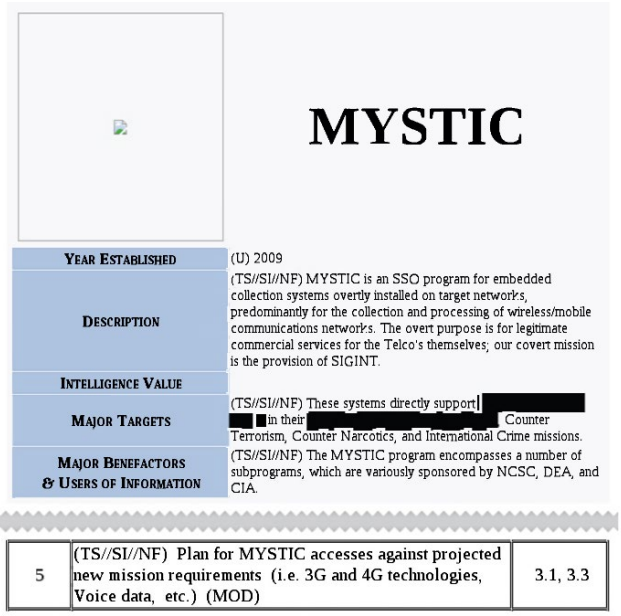
Teknisesti kahdentaminen tapahtuu taivuttamalla valokuitua siten, että pieni osa valosta karkaa vakoilulaitteiston vastaanottimeen. Tällöin päästään käsiksi paitsi viestiliikenteen metadataan ja salaamattomaan sisältöön, myös potentiaalisesti niin sanotusti naiivisti salattuun sisältöön, jonka symmetristä sessioavainta suojaa kättelyssä staattinen RSA-avain.

Salauksen purkaminen

GCHQ:n Broad Oak -ohjelma kerää tilastoja internetin yleisimmän salauksen, TLS:n (entinen SSL), trendeistä kuten avaintenvaihtoalgoritmien suhteellisista määristä. Sen mukaan vuonna 2012 RSA:n osuus oli 90, DHE:n 5 ja ECDHE:n 5 prosenttia. Jälkimmäiset avaintenvaihtoprotokollat ovat Diffie-Hellman Ephemeral sekä Elliptic Curve DHE. Näiden toiminta perustuu diskreetiksi logaritmiksi kutsuttuun matemaattiseen ongelmaan, jolla on niin kutsuttu perfect forward secrecy (PFS) -ominaisuus, joka estää Upstreamilla kerätyn liikenteen takautuvan ja passiivisen purkamisen.

RSA:n yleisyys on vakava ongelma. Jos FVEY:llä olisi pääsy palveluntarjoajien yksityisiin RSA-avaimiin, Upstream pystyisi purkamaan jopa 90 prosenttia maailman salatusta verkkoliikenteestä.

Todiste siitä, että avaimia vaaditaan, on U.S. vs Lavabit -oikeudenkäynti, jossa liittovaltio vaati National Security Letter -kirjeen nojalla Ladar Levisonia luovuttamaan yrityksensä yksityisen RSA-avaimen siitä huolimatta, että näin toimiminen vaarantaisi satojen tuhansien käyttäjien yksityisyyden. Lavabit on tiedettävästi ensimmäinen yritys joka teki periaatepäätöksen ennemmin sulkea palvelu, kuin vaarantaa käyttäjien yksityisyys. NSL oli samalla myös vaitiolomääräys, eli ny-



MYSTIC

YEAR ESTABLISHED	(U) 2009
DESCRIPTION	(TS//SI//NF) MYSTIC is an SSO program for embedded collection systems overtly installed on target networks, predominantly for the collection and processing of wireless/mobile communications networks. The overt purpose is for legitimate commercial services for the Telco's themselves; our covert mission is the provision of SIGINT.
INTELLIGENCE VALUE	
MAJOR TARGETS	(TS//SI//NF) These systems directly support [redacted] in their [redacted] Counter Terrorism, Counter Narcotics, and International Crime missions.
MAJOR BENEFACTORS & USERS OF INFORMATION	(TS//SI//NF) The MYSTIC program encompasses a number of subprograms, which are variously sponsored by NCSC, DEA, and CIA.

5	(TS//SI//NF) Plan for MYSTIC accesses against projected new mission requirements (i.e. 3G and 4G technologies, Voice data, etc.) (MOD)	3.1, 3.3

kyiset verkkoyritykset eivät ole pystyneet kertomaan, mikäli myös niiltä on vaadittu yksityisiä avaimia.

NSA:n Bullrun-ohjelma mahdollistaa erilaisten yleisten salausprotokollien kuten TLS:n, SSH:n sekä joidenkin VPN-, VoIP- ja pikaviestiprotokollien purkamisen. Menetelmät ovat ECI-luokiteltua tietoa, ja tiedusteluyhteisön sisällä jopa niistä spekulointi on kielletty. Asiakirjat mainitsevat kuitenkin, että menetelmät pohjautuvat päätelaitehyökkäyksiin (Computer Network Exploit tai CNE), postipalveluiden välittämien pakettien peukalointiin (interdiction), suhteisiin alan yritysten kanssa sekä matemaattisiin menetelmiin.

Asiantuntijat ovat arvioineet matemaattisten menetelmien liittyvän joko symmetriseen RC4-salausalgoritmiin tai RSA:n 1024-bittiseen versioon. Arviolta puolet TLS-yhteyksistä käyttää edelleen RC4:ää, ja suurin osa RSA-avaimista on ilmeisesti edelleen 1024-bittisiä. Lisäksi valvonta on saattanut hyödyntää kryptosodan jälkeen palvelimiin jätettyä tukea vientirajoitusten aikaisille algoritmeille, joihin yhteydet on voitu pudottaa.

Vaikka modernit sertifikaatit käyttävätkin laskennallisesti murtamattomia RSA-2048/4096- ja AES-algoritmeja, CNE:n maininta viittaa siihen, että yksityisiä RSA-avaimia myös varastetaan palvelimilta. Laskennallinen murtolujuus tai palvelimen sijoitus FVEY:n juridisen toimivallan ulkopuolelle ei siis ole minkäänlainen tae turvallisuudesta.

Luottamusketjussa on myös heikompi lenkki. Snowden-asiakirjoihin perehtynyt tietoturvatutkija ja toimittaja Jacob Appelbaum kertoo LibrePlanetin avauspuheessaan NSA:lla olevan yhteyksiä sertifikaatteja myöntäviin yrityksiin (CA). Asian vakavuutta kuvaa se, että jopa 84 prosenttia kaikista maailman sertifikaateista allekirjoitetaan Yhdysvalloissa. CA-yritysten yksityisillä avaimilla on mahdollista luoda väärennettyjä sertifikaatteja, eli niitä ei tarvitse varastaa palvelimilta. Koska NSA pystyy Quantumsquirrel-ohjelmalla runkoverkosta käsin naamioitumaan miksi tahansa verkkopalveluksi, se pystyy mies-välissä-hyökkäyksen avulla purkamaan melkein minkä tahansa salatun yhteyden www-palvelimelle.

Matkapuhelinten vakoilu

Keskitetyt avainlähteet ovat houkutteleva kohde. Asiakirjat osoittivat NSA:n varastaneen Gemalto-yritykseltä sen valmistamien SIM-korttien salausavaimia. Operaattorit katsoivat vakoilun edellyttävän lähiympäristössä toimimista, mutta koska operaattorin järjestelmät ovat kytköksissä Internetiin, ohjelmistoimplantit löytävät tiensä verkosta operaattorin järjestelmiin.

Vuodoista käy ilmi, että GCHQ:n Dishfire sieppaa päivittäin satoja miljoonia tekstiviestejä ympäri maailmaa, ja NSA:n Mystic-ohjelman alainen Somalge kykenee sekä keräämään että tallentamaan koko Bahamasaarten ja Afganistanin puheluliikenteen.

Mystic-järjestelmä vastaa teleoperaattoreiden signaalitiedustelusta.

Ratkaisuksi tietomurtoon on ehdotettu SIM-korttien vaihtoa ja turvallisten salausavainten levityskanavien muodostamista. Ongelma on kuitenkin se, että eri operaattorien jokainen SIM-kortti käyttää samaa salausavainta. Niiden toteutus on ollut alun alkaenkin teknisesti huono, joten SIM-korttien vaihtaminen olisi lähinnä purkkaratkaisu, joka loisi valheellisen turvallisuuden tunteen.

NSA pystyisi näet edelleen varastamaan salausavaimen joko operaattorilta tai suoraan SIM-kortista. NSA:n asiakirjojen mukaan SIM-kortin tavoin yleiskäyttöinen GID-avain on pystytty irrottamaan Applen A4-prosessorista. Lisäksi myös tietokoneiden TPM-siruista on kyetty hakemaan henkilökohtaisia salausavaimia.

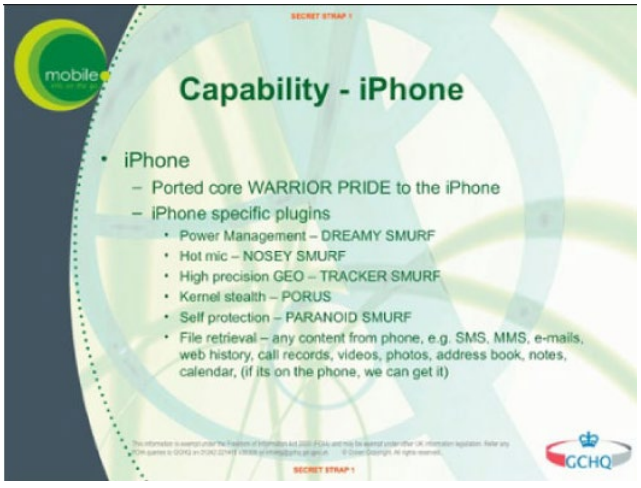
NSA on luultavasti maailman edistynein taho, mitä tulee differentiaalisen jänniteanalyysin kaltaisiin kryptanalyyttisiin sivukanavahyökkäyksiin. Vaihtoehtoisesti NSA voisi hioa läpi SIM-korttien kryptoprosessorin kerros kerrokselta ja kaivaa salausavaimen sirusta muun muassa tunnelielektronimikroskoopilla.

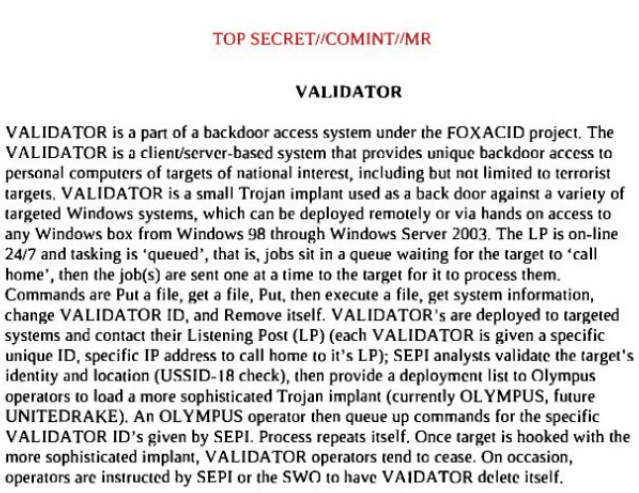
Päätelaitehyökkäykset

Snowden-asiakirjojen mukaan NSA:lla on käytössään välineitä, joilla se voi tunkeutua myös älypuhelimien kaltaisiin päätelaitteisiin. Tämän myötä monet yritykset, kuten Apple, ovat ilmoittaneet salaavansa jatkossa puhelinten tiedot oletusarvoisesti. Tästä ei kuitenkaan ole apua, sillä päätelaitteen salaus toimii vain datan ollessa levossa eli puhelimen ollessa pois päältä. Salausavain on käynnissä olevan puhelimen työmuistissa, johon Kanadan tiedustelupalvelu CSECin Warrior Pride sekä siihen pohjautuva GCHQ:n Daredevil-ohjelmisto pääsevät verkon kautta käsiksi.

GCHQ:n älypuhelimiin kohdistettu CNE-järjestelmä.

Ohjelmisto kykenee myös hallitsemaan puhelimen käynnistymistä ja mikrofonia sekä varastamaan kaiken

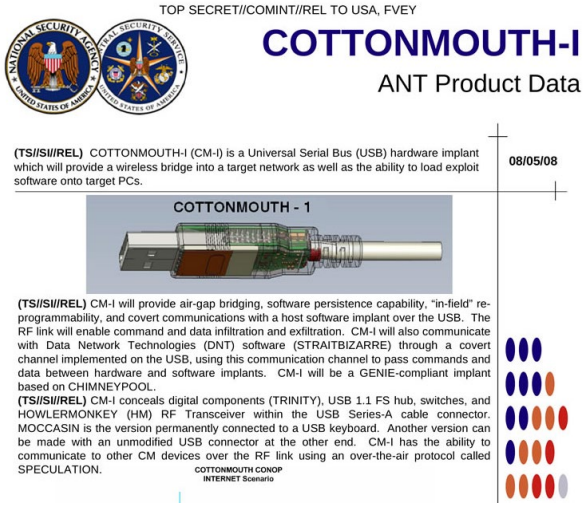




TOP SECRET//COMINT//MR

VALIDATOR

VALIDATOR is a part of a backdoor access system under the FOXACID project. The VALIDATOR is a client/server-based system that provides unique backdoor access to personal computers of targets of national interest, including but not limited to terrorist targets. VALIDATOR is a small Trojan implant used as a back door against a variety of targeted Windows systems, which can be deployed remotely or via hands on access to any Windows box from Windows 98 through Windows Server 2003. The LP is on-line 24/7 and tasking is 'queued', that is, jobs sit in a queue waiting for the target to 'call home', then the job(s) are sent one at a time to the target for it to process them. Commands are Put a file, get a file, Put, then execute a file, get system information, change VALIDATOR ID, and Remove itself. VALIDATOR's are deployed to targeted systems and contact their Listening Post (LP) (each VALIDATOR is given a specific unique ID, specific IP address to call home to it's LP); SEPI analysts validate the target's identity and location (USSID-18 check), then provide a deployment list to Olympus operators to load a more sophisticated Trojan implant (currently OLYMPUS, future UNITEDRAKE). An OLYMPUS operator then queue up commands for the specific VALIDATOR ID's given by SEPI. Process repeats itself. Once target is hooked with the more sophisticated implant, VALIDATOR operators tend to cease. On occasion, operators are instructed by SEPI or the SWO to have VAIDATOR delete itself.



puhelimeen tallennetun tiedon. Applen ilmoitus puhelinten salauksesta sai äänekästä vastustusta viranomaisilta, mutta näiden valitus on enemmänkin Applen markkinointia. Snowden-asiakirjojen näet arvioidaan maksavan lähivuosien aikana Yhdysvaltojen teknologiateollisuudelle kymmeniä miljardeja.

Päätelaitehyökkäykset kohdistuvat myös tietokoneita vastaan. Esimerkiksi XKeyscore-niminen internet-puskuri pystyy analysoimaan verkon yli kulkevia Windows-käyttöjärjestelmän automaattisia virheraportteja. Näistä virheraporteista voidaan tunnistaa käyttöjärjestelmässä tai sille asennetuissa ohjelmissa kuten selaimessa piileviä haavoittuvuuksia. Quantum-niminen järjestelmä kykenee runkoverkosta käsin kilpailemaan web-palvelimen lähettämän Server-Hello-paketin kanssa. Tämä mies vierellä (man-on-the-side) -niminen hyökkäys yhdistää käyttäjän NSA:n Foxacid-palvelimeen, joka pystyy ujuttamaan haittakoodia (exploit) käyttäjän saamaan verkkosivuun.

Vaihtoehtoisesti Quantum voi lisätä haittakoodin DNS-injektioon pohjautuvalla mies-välissä-hyökkäyksellä. Haittakoodi hyödyntää joko vanhentuneen selaimen tunnettua tai ajantasaisen selaimen vielä tuntematonta haavoittuvaisuutta ja kaappaa kohdekoneen haltuunsa.

Päätelaitteille tunkeutuminen on perinteisesti katsottu kohdistetuksi hyökkäykseksi, jolla ei ole mitään tekemistä massavalvonnan kanssa. NSA:n Quantumin Validator ja Commondeer edustavat kuitenkin ”untasked targeting” -valvontaa, eli XKeyscore pystyy hyökkäämään tietokoneita ja tietoverkkoja vastaan automaattisen avainsana-analyysin perusteella.

Automaattiseen tunkeutumiseen soveltuvan Validator-ohjelman kuvaus.

Kuten Appelbaum totesi, ”automatisoitu kohdistettu valvonta ei eroa skaalaltaan, ainoastaan menetelmältään.” Tietokoneelle tunkeutumisen jälkeen Foxacid pystyy välittämään modulaarisen Unitedrake-ohjelmakuorman (payload) kohdekoneeseen. Kuorman huomionarvoisia komponentteja ovat Salvagerabbit, joka varastaa järjestelmästä yksityisiä GPG-salausavaimia ja salasanatietokantoja tai luottamuksellisia asiakirjoja, sekä Grok, joka kerää näppäinpainallukset ja siten salasanat.

Käyttäjä ei pysty suojautumaan edes pitämällä järjestelmänsä ajan tasalla, sillä NSA ostaa vuosittain kymmenillä miljoonilla dollareilla ennalta tuntemattomia ja paikkaamattomia haavoittuvuuksia ja niitä hyväksikäyttäviä hyökkäysohjelmia (zero day exploit).

Haavoittuvuuksien tunnistaminen on tavallisen käyttäjän ulottumattomissa. On oletettavaa, että NSA:n haittaohjelmista ja järjestelmiin tunkeutumisesta vastaava osasto, Tailored Access Operations (TAO), osaa hioa näitä hyökkäysohjelmia, kunnes kaupalliset virustentorjunta- tai tunkeutumisenhavaintaohjelmat (IDS:t) eivät enää huomaa niitä. Tietoturvayritys Symantecin teettämä tutkimus osoitti, että yksittäisen korjaamattoman haavoittuvuuden keskimääräinen elinikä (window of exposure) on 312 vuorokautta. Tuntemattomia haavoittuvuuksia hyödyntäviä ’untasked targeting’ -hyökkäyksiä voidaan siis pitää seuraavan sukupolven massavalvontana.

Mikä sitten on kohdistettua ja mikä massavalvontaa? Erottelu ei ole oleellista, sillä valvonnan doktriini on ”kerää kaikki, pura kaikki, kräkkää kaikki.”

Voisikin sanoa, että valvontajärjestelmien rakenne vastaa nykymaailman järjestelmien tilaa, ja hankintatekniikka riippuu informaation suojauksesta. Kustannukset eivät ole ongelma, vaikka keräys olisikin välillä tappiollista. Riittää, että kaiken informaation kokonaisarvo vastaa hankintakustannuksia. Siinä missä laiteimplantin saaminen yksittäiseen verkosta eristettyyn järjestelmään maksaa kymmeniä tai satoja tuhansia dollareita, haittaohjelmalla on ainoastaan sen hankinta- tai kehityskustannus. Ohjelmistoimplantin kustannushyöty onkin sitä parempi, mitä laajemmin haavoittuvuutta hyödynnetään, eikä sen havaitsemisriski riipu niinkään kohteiden määrästä vaan näiden teknisestä osaamisesta.

30C3-konferenssin esityksessään ”To Protect and Infect – part 2” Appelbaum kertoo NSA:n ANT-katalogista, jossa on lueteltu jopa 50 erilaista hyökkäysimplanttia. Ohjelmistoimplanttien kappalehinnaksi on useimmiten merkitty nolla, mutta laitteiden hinnat yltävät jopa 50 000 dollariin. Laite-esimerkkeinä Cottonmouth ja Firewalk ovat USB-kaapeliin ja emolevyn USB-liittimeen piilotettuja radiolähettimiä, joilla päästään käsiksi internetistä irti kytkettyihin järjestelmiin ja lähiverkkoihin.

NSA:n ANT-tiimin rakentama Close Access Operations -laitteistoimplantti.

Ragemaster sekä Surlyspawn taas ovat näppäimistön ja näytön kaapeleihin piilotettavia retroreflektoreita.

Mikroaaltopulssilla valaistuna ne heijastavat takaisin joko näppäinlyönnit tai ruudulla näkyvän kuvan.

Nämä tieteiskirjallisuudelta kuulostavat keinot olivat vakoilutekniikan terävintä kärkeä jo vuosikymmen sitten. Implanttien kaltaiset tekniset menetelmät asettavat riman myös Suomen operaatioturvallisuudelle sekä julkiselle keskustelulle, nyt kun verkon puolustamis- ja hyökkäysmenetelmistä ollaan laatimassa strategiaa ja lainsäädäntöä.

Suomeen kaavailtu verkkovakoilu – uhka vai mahdottomuus

Suomen tiedustelulainsäädännön mietintötyöryhmän lainsäädäntöjohtaja Hanna Nordström kuvaili Talentumin haastattelussa Suomeen kaavaillun tietoliikennetiedustelun vastaavan NSA:ta siinä määrin, että menetelmät ovat ”mahdollisimman lieviä, mutta silti vielä tehokkaita”.

Aaltoyliopiston järjestämässä yleisöluentosarjassa korostettiin tosiasioiden tunnustamista. Kyberturvallisuuskeskuksen johtaja Kirsi Karlamaa välitti saamansa asiantuntija-arvion siitä, että ”aurinko ehtisi sammua ennen kuin internetin kaikki kryptaus ehdittäisiin purkaa”. Väite pitää paikkansa mutta on vain puolitotuus: aurinko ehtisi sammua jo ennen, kuin koko planeetan nykyinen ja tuleva laskentakyky ehtisi purkaa ensimmäistäkään nykyaikaisesti salattua viestiä.

Moderneja salausalgoritmeja ei murreta, vaan ne kierretään. Kiinnostava informaatio tuskin liikkuu suomalaisissa verkkopalveluissa. Käytössä pitäisi siis olla Prismin kaltainen kanava ulkomaisille palvelimille, tai sitten salausavaimia pitäisi hankkia joko §702 FISA:n kaltaisella laillisella tai Bullrun-ohjelman kaltaisella laittomalla menetelmällä.

Suomen viranomaisten tekninen kyky päästä käsiksi ulkomaista salattua palvelua käyttävän ISIS-solun kommunikaatioon ilman FVEY:n apua on käytännössä nolla. Runkoverkon vakoilusta ei ole konkreettista hyötyä edes metadatan suhteen, sillä Tor-verkon läpi itsensä anonymisoiva, Lähi-itään kommunikoiva ISIS-solu ei erottuisi runkoverkossa muista Tor-käyttäjistä mitenkään.

Suomen runkoverkosta tietoa onkisikin tehokkaimmin NSA, mutta vieraan vallan vakoilulaitteiden kytkeminen runkoverkkoon jakanee mielipiteet. Onko tilanne, jossa Yhdysvallat vakoilee koko Suomen sisäistä tietoliikennettä, mahdollisimman vähän yksityisyyttä loukkaava?

Keskeisempi ongelma tietoliikennetiedustelulle on kuitenkin se, etteivät yksityisyydestään huolehtivat kansalaiset (ja terroristit) luota valtarakenteille alisteisiin sertifikaatteihin. Sen sijaan he hallitsevat salausavaimiaan itsenäisesti käyttämällä työkaluja, joiden protokolla (kuten PGP, OTR tai ZRTP) on päästä päähän salattu (end-to-end encrypted tai E2EE).

Ainoa tehokas tapa seuloa päästä päähän salatusti kommunikoiva ääriliikesolu Suomesta olisi tunkeutua jokaiseen näitä protokollia käyttävään tietokoneeseen ja älypuhelimeen, varastaa yksityiset salausavaimet ja niitä suojaavat salasanat, sekä ZRTP- ja OTR-viestinnän tapauksessa vielä hyökätä aktiivisesti jokaista erillistä keskustelua vastaan. Vasta sitten voitaisiin purkaa ja lukea liikenne ja selvittää onko kohde terroristi.

Seulonta on siis teknisesti toteutettavissa, mutta ensin olisi syytä käydä pitkä julkinen keskustelu siitä, jättääkö toimenpiteillä saavutettu turvallisuus mitään turvaamisen arvoista, jos Suomen ihmisoikeustilanne vajoaa Venäjän ja Kiinan alapuolelle.

Ratkaisuja etsimässä

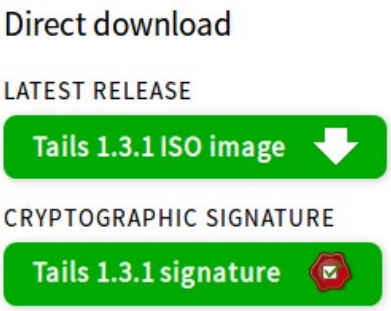
Viestijärjestelmien tietoturvan parannus vähentää niiden helppokäyttöisyyttä. Pahimmillaan joko tietoturvan kannalta oleelliset varmistukset jäävät tekemättä, tai sitten siirrytään helpompaan sovellukseen, joka ei tarjoa niitä lainkaan.

Tietoturva katsotaan monesti suhteelliseksi: kaikki valvontatekniikat eivät kosketa kaikkia, joten vähäiselläkin suojauksella voi saada konkreettista tietoturvaa. Toisaalta tietoturva on myös absoluuttista, sillä valtiollisen toimijan tavoite on saada hyökkäysmenetelmät skaalautumaan automaattisesti suojauskeinojen mukaan. Silloin mahdolliset lisäkustannukset jäävät tulosvastuuttomuuden ja massiivisen budjetin varjoon.

Tietoturvasta voidaan siis puhua vasta, kun viestintä on valvontateknologian ulottumattomissa, ja valitettavasti se edellyttää työkalujen käytön opettelua. Verkkovakoilun tekninen torjunta edellyttää radikaalia suunnanmuutosta niin työkaluissa kuin käyttäjien asenteissa.

Ensimmäinen askel on luottamuksen hajauttaminen. Juridisesti haavoittuva, keskitetty julkisen avaimen infrastruktuuri (PKI) on ollut harha-askel. Välttämätön osa tietoturvan alustusta on rakentaa luja luottamuksen verkko (Web of Trust), jossa digitaalisista allekirjoitusketjuista huolehtivat yritysten sijaan vapaat yksilöt.

Esimerkiksi Tor Projectin selain Tor Browser Bundle ja tietoturvaa tarjoavat Tails- ja OpenBSD-käyttöjärjestelmät allekirjoitetaan jo PGP:llä, mutta julkisten avainten autenttisuudesta varmistuminen edellyttää globaalia verkostoitumista. Tässä ovat avainasemassa sähköisiä oikeuksia puolustavat järjestöt.



Tails-kehitystiimi suosittaa tarkistamaan levykuvan PGP-allekirjoituksen.

Luottamuksen verkko mahdollistaa turvallisen sovellusten välittämisen, mutta myös itse sovelluksia on muutettava. Tarvitaan ratkaisuja, jotka ovat helppokäyttöisiä ja tietoturvallisia siten, ettei niiden murtaminen ole automatisoitavissa ja skaalattavissa ilman lineaarisesti kasvavia kustannuksia. Arvostettu kryptografi Matthew Green kritisoi nykyisiä päästä päähän salattuja ohjelmistoja, sillä verkkoon kytkettyyn päätelaitteeseen hyökkääminen tekee niistä turvattomia.

Viestinnän salaus, allekirjoitus, autentikoiminen ja purkaminen on siirrettävä verkosta erotettuun ympäristöön, luotettavan tietojenkäsittelyn alustaan (Trusted Computing Base / TCB). Alustoja suunniteltaessa on tärkeää välttää sokeaa luottamusta omisteiseen laitesuunnitteluun ja ohjelmakoodiin sekä ulkomaisille viranomaisille alisteisiin yrityksiin. Avoimet laitteet voitaisiin valmistaa jopa Suo-

messa: esimerkiksi VTT:llä on mikroelektroniikan valmistuslinja.

Lainaten kuuluisaa kryptografia ja tietoturva-alan asiantuntijaa Bruce Schneieria, ”yhteiskunnassa on valittava joko globaalisti tietoturvattomat ja valvotut tai turvalliset ja yksityiset järjestelmät.”

Teknologistit puoltavat jälkimmäistä yleisen turvallisuuden, YK ihmisoikeuksien ja ekonomistit globaalin kilpailukyvyn säilyttämisen näkökulmasta. Vahvoja ratkaisuja, kuten Genode, Crash-safe sekä allekirjoittaneen TFC, on jo kehitteillä. Niiden käytettävyyden parantumista odotellessa nopein ensiapu on tehdä nykyisestä passiivisesta ja julkisia avaimia hyödyntävästä massavalvonnasta taloudellisesti kannattamatonta.

Viestinnän sisältöä voidaan suojata päästä päähän salatuilla sovelluksilla, ja metadatan määrä voidaan minimoida reitittämällä liikenne Tor-verkon läpi. Sovellusten tarkemmalle tarkastelulle ei tässä ole tilaa, mutta kelpaamattoman tunnistaa Cypherpunk-yhteisön hyväksymillä ”Stefin seitsemällä säännöllä tunnistaa käärmeöljy”:

1. suljettu lähdekoodi
2. selaimessa suoritettava ohjelma
3. mobiilisovellus
4. käyttäjän kyvyttömyys kontrolloida salausavaimia
5. uhkamallin puute
6. markkinointitermit kuten ”kyber” tai ”sotilasluokan salaus”
7. välinpitämätön asenne päätelaitteiden tietoturvattomuutta kohtaan.

Suositeltavia ohjelmia

Käyttöjärjestelmäksi on erittäin suositeltavaa vaihtaa joko GNU/Linux tai jokin BSD-variantti, tai ainakin asentaa se omisteisen Windows- tai OSX-käyttöjärjestelmän rinnalle. Applen, Googlen ja Microsoftin käyttöjärjestelmät edellyttävät sokeaa luottamusta siihen, ettei Yhdysvallat ole vaikenemismääräyksen kera pakottanut yrityksiä lisäämään niihin takaportteja tai viivyttämään niistä löydettyjen haavoittuvuuksien paikkaamista. Lisäksi suljettu lähdekoodi tekee haavoittuvuuksien paikantamisesta käytännössä erittäin vaikeaa. Vaikkei käyttöjärjestelmää olisikaan valmis vaihtamaan, on avoimen lähdekoodin ohjelmien käyttö Windowsilla tai OSX:llä silti askel parempaan suuntaan.

Pilvipalveluiden sijaan kannattaa suosia kotipalvelinta, jonne tiedostot varmuuskopioi. Jos tähän ei ole mahdollisuutta, on palveluksi valittava sellainen, jonka asiakasohjelma on avointa lähdekoodia ja joka säilyttää salausavainta paikallisesti. Yksi harvoista nämä vaatimukset täyttävistä palveluista on Owncloud.

Sähköpostin lähettämiseen suositellaan joko Claws Mail – tai Mozilla Thunderbird -sovelluksia. Molemmat ovat vapaita ohjelmia, tukevat lukuisia käyttöjärjestelmiä ja liitännäisten kanssa PGP-salausta.

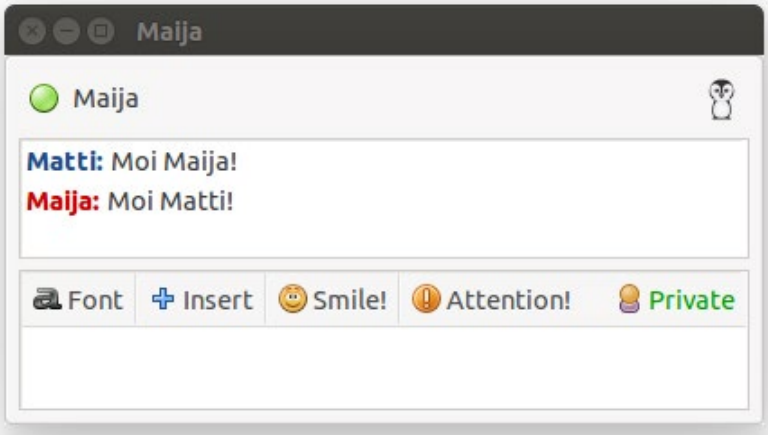
Pikaviestintään sopivat esimerkiksi OTR-protokollaa tukevat avoimen lähdekoodin sovellukset kuten Jitsi tai Adium. Pidgin soveltuu tähän myös, joskin OTR-laajennus on asennettava erikseen.

Älypuhelimille toimivin puhe- ja viestintäsovellus on iOS:n Signal, joka toimii yhteen Androidin TextSecure- ja Redphone-sovellusten kanssa. Kaikki kolme ovat ammattimaisesti tehtyjä, ilmaisia, vapaita ja päästä päähän salattuja. Älypuhelinten ongelma on kuitenkin, että niiden sovellusprosessorit ovat alisteisia niin sanotulle Baseband-prosessorille, jolla ajetaan suljettua lähdekoodia. On siis vaikea sanoa, onko puhelinten ohjelmakoodissa haavoittuvuuksia tai takaportteja. Kuten Snowden sanoi, ”hyvin rahoitettu tiedustelupalvelu pystyy murtautumaan puhelimeen sillä hetkellä kun se kytketään päälle”, mutta koska ihmiset viestivät joka tapauksessa liikkeellä ollessaan, on suotavaa vaikeuttaa verkkovalvontaa niin paljon kuin on teknisesti mahdollista.

Paras tunnettu AES-256 algoritmin heikkoutta hyödyntävä Biclique-hyökkäys (https://en.wikipedia.org/wiki/Biclique_attack) tekee murtamisesta 21,6 = 3,03 kertaa nopeamman.

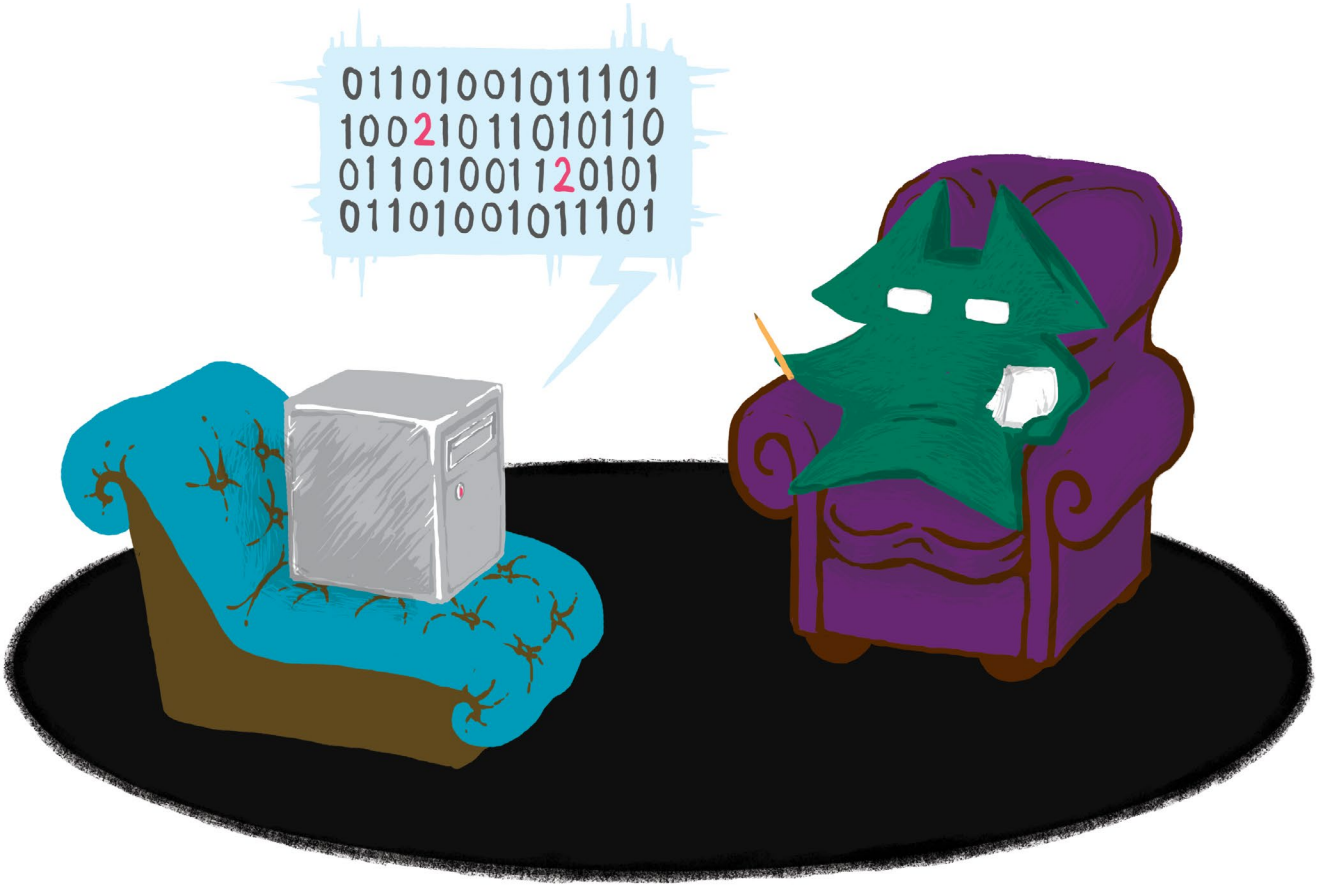
Edes kvanttitietokone ei murra AES-256:ta: Groverin algoritmi (https://en.wikipedia.org/wiki/Grover%27s_algorithm) käsittelisi AES-256:n samaan tapaan kuin tavallinen tietokone käsittelisi AES-128-algoritmin: laskenta olisi käsittättömästi nopeampaa mutta silti veisi kauemmin kuin planeetallamme on elinaikaa jäljellä.

Kaikki salausalgoritmit eivät tietenkään ole yhdenvertaisia, mutta yksikään nykyaikainen verkkoyhteys tai kommunikointiohjelma ei tänä päivänä käytä heikkoja algoritmeja kuten IDEA tai RC4.



Participants: matti@jabber.org, maija@jabber.org
Message Count: 2
Date: Mar 29, 2015 15:04:22 PM
Duration: <1 minute

[15:04 PM] Matti: ?
OTR:AAIDAAAAAAMAAAADAAAAwIYs2awYHJzQHlRN3T4H1sAiG3pw6GmIpCh6Q0+LNHbEwxloOFwItuUECn5kSbRFi9ELAn/BHWIzjk+djfWwo+u45R3JpF0tTY4Fn1aW4vxyhrTgs/4Uer1a9S93JVyCBrK/jZWMGK0ac1FZ8hGicoZyvk4xEQ4P1AjSaNIRmkhVcIWF7AEIN7tpZKGY5Htk8vtvTuwdZYwQU2p+hHozzhgC91A9z+hRpaQISXaqcEXRQtJlfdV+hV0cs9ROkKd6bAAAAAAAAAABAAAABvoLfJi3QLsUE55c9XT5TWn3KLiUwqDxPMq6AAAAFKPGzbIV/vhzmi1XQ5aKzBNCsAoc.
[15:04 PM] Maija: ?
OTR:AAIDAAAAAAMAAAAEAAAAwK+qQNaMFvkUUc3G/QeR6FySppaaZyj2UXNeCBdbCXF6B3Yg3z9XtuddA8kQejuK/0YcYBqMiPM4ZL8hUK8vmcckA+Qb0ffZGKaAck0IAer6BSZKNfQDZEn85cO2LVBFuOsRCwoIHCHadOOxqKey57t6LjVMgZRcAoMpWgv4c0CFAeVKMPTFQ9G5rAQoqdBYdgxqkUiL/Olsfg35DIFGLiNHxh4XI5Or19MGhmg0rIeCHnViN/qYRVNZPvdPozIw5wAAAAAAAAABAAAABvdYEH6XhTuYPKFubSxpMuPwfrm+lL8dGwlBAAAAFIPH644Yp0qFyKa6mAEWuuMi7lec.



Konekielen kiemurat: Portti tietokoneen sielunelämään

Konekieli on aina kuulostanut hohdokkaalta tietokoneharrastajan korvaan – onhan se laitteenläheisin taso, jolle ohjelmoija pääsee. Vaikka konekieli ei olekaan enää portti taianomaisiin ohjelmointitemppuihin, auttaa sen ymmärtäminen edelleen käsittämään tekniikkaa.

Teksti: Ville-Matias Heikkilä

Kuvat: Mitol Meerna, Ville Matias Heikkilä, Visual6502.org, AMD

Aiemmin konekieliohjelmointi oli liki pakollinen taito esimerkiksi peli- tai demo-ohjelmoijalle, mutta nykyisin konekieltä tuottavat lähinnä korkean tason kielten kääntäjät. Konekielen lukutaidosta on silti edelleen hyötyä: kääntäjän työtä pystyy arvioimaan ja lähdekoodittomia ohjelmia tutkimaan ja muokkaamaan. Niin kone kuin sen ohjelmatkin muuttuvat taidon myötä paljon käsinkosketeltavammiksi. Vanhojen koneiden, mikrokontrollerien ja matalan tason tietoturva-aukkojen parissa askarteleville konekieli on edelleen tärkeä väline.

Konekieliä on monia

Kaikki koneet eivät suinkaan ymmärrä samaa konekieltä. Esimerkiksi PC-suorittimet käyttävät X86-konekieltä ja mobiililaitteet useimmiten ARM-konekieltä. Yksittäistä konekieltä kutsutaan myös käskykannaksi tai arkkitehtuuriksi.

Tässä artikkelissa keskitytään selkeyden vuoksi neljään käskykantaan mikrotietotekniikan historian varrelta: 6502, X86, 68K ja ARM. Koska nämä käskykannat eroavat melkoisesti suunnittelufilosofioiltaan, saa niihin tutustumalla myös yleiskäsityksen siitä, millaisia konekieliä on olemassa.

MOS Technologyn 6502 on ollut suosituimpia 8-bittisiä suorittimia. Sitä lähisukulaisineen ja klooneineen ovat käyttäneet esimerkiksi Applen, Atarin ja Commodoren 8-bittiset koneet sekä Nintendon NES. 6502:n perinteinen kilpailija on ollut Zilogin Z80, joka perustuu Intelin 8080:aan.

Eri konekielten käskyjä biteiksi purettuna.

64-bittisen nyky-X86:n rekisteristön vanhimmat osat ovat 70-luvulta.

Uudempia, lähinnä sulautetuissa järjestelmissä käytettyjä 8-bittisiä käskykantoja ovat AVR ja PIC.

Intelin X86 on tuttu IBM PC -yhteensopivista koneista. Alun perin 16-bittistä käskykantaa on laajennettu ja uudistettu sittemmin radikaalisti – 386-suorittimessa 32-bittiseksi ja 2000-luvulla AMD:n aloitteesta 64-bittiseksi. Uudistuksista huolimatta historian eri kerrokset näkyvät edelleen hyvin läpi X86-konekielessä.

Motorolan 68K:ta käytti suurin osa PC:n kilpailijoista 90-luvun alkupuolelle asti: Amiga, Atari ST, Macintosh, Unix-työasemat. Se perustuu 70-luvun isompien tietokoneiden käskykantoihin ja on suunnittelultaan tyylipuhdas CISC (Complex Instruction Set Computer).

ARM on tämän hetken suosituin käskykanta. Se dominoi etenkin mobiililaitteita mutta saattaa ennen pitkää syrjäyttää myös X86:n. Alkujaan Archimedes-kotimikrossa käytetty käskykanta nousi suosioon, koska se tarjosi paljon tehoa vähäisellä piimäärällä. ARM on RISC-käskykanta (Reduced Instruction Set Computer). Muita RISCejä ovat esimerkiksi MIPS, SPARC, PowerPC ja AVR.

Käskyn olemus

Konekielikäskyt ovat melko tiivisrakenteisia ykkösten ja nollien jonoja. Ohessa on esitetty saman tehtävän suorittava käsky usealla eri konekielellä. Kukin käsky summaa luvun 3 johonkin suorittimen sisäiseen rekisteriin, mutta esimerkiksi bittileveys vaihtelee: 6502:n `adc` laskee 8-bittisillä luvuilla, joiden summat voivat olla enintään muutamia satoja, kun taas ARMin 32 bittiä leveä toimitus yltää miljardeihin asti. Suorittimen tai käskykannan bittisyys tarkoittaa yleensä nimenomaan peruslaskutoimitusten suurinta bittileveyttä.

Ykkösten ja nollien jonot ovat ihmisille vaikeaselkoisia. Ihmiset käsittelevät konekieltä tämän vuoksi yleensä symbolisessa eli assembly-muodossa. Assembly-esityksestä voi myös arvata, mitä käsky tekee, vaikka käskykanta olisi ennestään tuntematon: esimerkiksi alkukirjaimet `ad(d)` viittaavat yhteenlaskuun. Samalla konekielellä voi olla useita erilaisia assembly-kielimuotoja, joita eri assembly-kääntäjät eli assemblerit käyttävät – esimerkiksi X86:lla Intel- ja AT&T-syntaksit.

Konekielikäsky jakautuu yleensä operaatiokoodiin, osoitusmuotoon ja operandeihin. Operaatio on ”verbi”, jota vastaa assemblyssä ensimmäinen sana, jota kutsutaan muistikaaksi (mnemonic), esimerkiksi `add`. Operandit ovat sen jäljessä tulevia ”substantiiveja”: rekistereitä, lukuja ja muistiosoituksia. Osoitusmuotoja taasen voi verrata ihmiskielten taivutusmuotoihin. Ne ilmaisevat, miten operandiosa tulkitaan – onko kyseessä esimerkiksi muistiosoite vai suora luku – ja antavat sille mahdollisia lisämääreitä: esimerkiksi 68K-käskyn pääte `.b`, `.w` tai `.l` ilmaisee, suoritetaanko toimitus 8-, 16- vai 32-bittisenä.

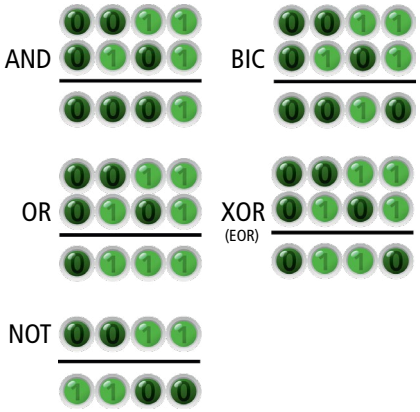
Data pyörii rekistereissä

Useimmissa konekielissä suurin osa datankäsittelystä tapahtuu rekistereissä, jotka voi ajatella suorittimensisäisiksi kiinteiksi muuttujiksi. Rekisterien määrät, leveydet ja käyttötavat vaihtelevat huomattavasti käskykannasta toiseen.

6502:n rekisteristö on hyvin pieni ja kukin rekisteri on sidottu tiettyihin tehtäviin. Suurin osa laskutoimituksissa on pakko tehdä akkurekisterissä eli A:ssa. Indeksirekisterit X ja Y soveltuvat lähinnä muistiosoitukseen ja silmukkalaskureiksi, mihin taas A ei sovellu. Näiden lisäksi 6502:ssa on vain pino-osoitin S, statusrekisteri P ja seuraavan käskyn muistiosoitteen ilmoittava käskyosoitin PC. PC on ainoa 16-bittinen rekisteri, muut ovat 8-bittisiä. Suppeaa rekisteriavaruutta täydentää niinsanottu nollasivu eli muistin 256 ensimmäistä tavua, ja monet muistiosoitustavat onnistuvatkin vain nollasivun kautta.

ARMin ja muiden riscien rekisteristö on puolestaan hyvin symmetrinen ja yleiskäyttöinen. Periaatteessa mitä tahansa rekisteriä voi käyttää mihin tahansa. Ainoat poikkeukset ovat rekisteri R15 eli käskyosoitin sekä erillinen statusrekisteri. Perus-ARMissa on 32-bittisiä rekistereitä 16, mutta useimmissa muissa risceissä perusrekistereitä on 32 tai enemmän.

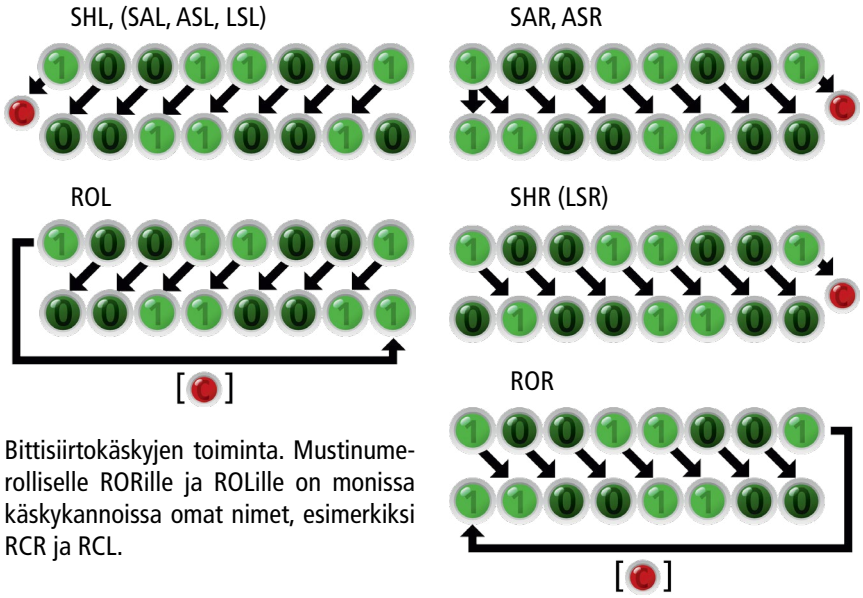
X86:n rekisterit ovat alkujaan erikoistuneita: esimerkiksi muistiosoitukseen on voinut käyttää vain rekistereitä BX, SI, DI ja BP. 32-bittinen päivitys kuitenkin löyhensi rajoituksia. Nykyisessä 64-bittisessä toimintamoodissakin on silti edelleen jäljellä käskyjä, jotka on sidottu tiettyihin rekistereihin: esimerkiksi yksitavuinen käsky `stosb` tallentaa 8-bittisen AL-rekisterin (*accumulator low*) sisällön muistipaikkaan, johon alkuperäisen DI-rekisterin (*destination*



Tämän artikkelin käskykantojen bitittäiset operaatiot. BIC on käytössä ARMissa.

	Intel-X86	68K	AT&T-X86
Operandijärjestys	add kohde,lähde	add.w lähde,kohde	addw lähde,kohde
Muistiosoitus	add ax,[1234]	add.w 1234,kohde	addw 1234,%ax
Välitön luku	add ax,1234	add.w #1234,kohde	addw $1234,%ax
Indeksoitu osoitus	[ebx+esi+8]	8(a0,d1.L)	8(%ebx,%esi)
Heksadesimaali	1234h	$1234	0x1234
Käskyn oma sijainti	jmp $	jmp pc	jmp .
Datatavu	db 123	ds.b 123	.byte 123

Assembly-syntaksit ovat yleensä melko samankaltaisia, mutta niiden välillä on myös hämääviä eroja. Tässä muutamia esimerkkejä.



Bittisiirtokäskyjen toiminta. Mustinumerolliselle RORille ja ROLille on monissa käskykannoissa omat nimet, esimerkiksi RCR ja RCL.

8×	4×	2×	1×	Etumerkitön	Etumerkillinen
0	0	0	0	0	+0
0	0	0	1	1	+1
0	0	1	0	2	+2
0	0	1	1	3	+3
0	1	0	0	4	+4
0	1	0	1	5	+5
0	1	1	0	6	+6
0	1	1	1	7	+7
1	0	0	0	8	-8
1	0	0	1	9	-7
1	0	1	0	10	-6
1	0	1	1	11	-5
1	1	0	0	12	-4
1	1	0	1	13	-3
1	1	1	0	14	-2
1	1	1	1	15	-1

Nelibittiset kokonaisluvut tulkittuna etumerkittöminä ja etumerkillisinä kahden komplementtia käyttäen.

index) 64-bittiseksi laajennettu versio RDI osoittaa.

68K:n perusrekisteristö on jaettu kahdeksaan data- ja osoiterekisteriin D0-D7 ja A0-A7, joista A7:ää käytetään pino-osoittimena. Erikseen on vielä statusrekisteri CCR ja käskyosoitin PC. Osoiterekisterit ovat alun perin 24-bittisiä mutta laajennettiin 68020-suorittimessa 32 bittiin. Kaikilla rekistereillä voi suorittaa laskutoimituksia melko yleiskäyttöisesti, mutta muistiosoitukset on hoidettava osoiterekistereillä.

Käskyt taipuvat osoitusmuodoissa

Rakenteeltaan yksinkertaisimmat konekäskyt eivät saa operandeja lainkaan, joten niiden toiminta on sidottu tiettyihin rekistereihin. Edellä mainittu X86:n `stosb` on esimerkki tästä niinsanotusta implisiittisestä osoitusmuodosta. Muita esimerkkejä ovat aliohjelmasta palaavat käskyt (`ret`, `rts`) ja lippujen nosto- ja laskukäskyt (`sec`, `clc`).

Käskyn tyypillinen operandimäärä vaihtelee konekielestä toiseen. 6502:ssa useimmat käskyt ovat yksioperandisia. Tämä operandi on yleensä muistiosoitus, jolloin laskutoimitus tapahtuu akkurekisterin ja muistipaikan välillä. X86 ja 68K ovat kaksioperandisia, eli käskyyn kuuluu lähde- ja kohdeoperandi. Tyypilliseen ARM-käskyyn puolestaan kuuluu kolme operandia – kaksi lähdettä ja yksi kohde. Nollaoperandisina konekielinä voidaan pitää Forth-tyylisiä pinopohjaisia konekieliä.

Useimmilla suorittimilla valtaosa konekielikoodista on rekisterienvälistä pyörittelyä. Operandeina voi kuitenkin käyttää rekisterien lisäksi myös välittömiä lukuja (*immediate*) tai erilaisia muistiviittauksia.

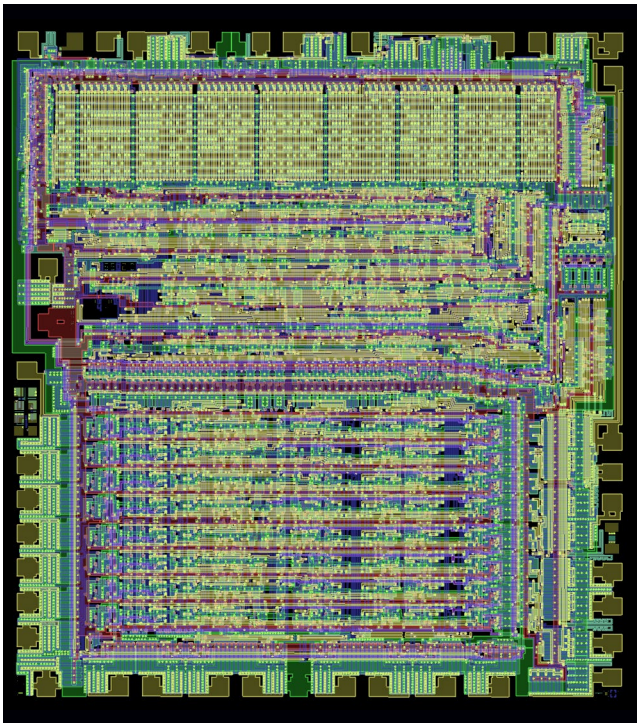
Operandien yhdistämisessä on usein

```
clc         asl $FE
lda $FE     rol $FF
adc #$34    asl $FE
sta $FE     rol $FF
lda $FF     asl $FE
adc #$12    rol $FF
sta $FF
```

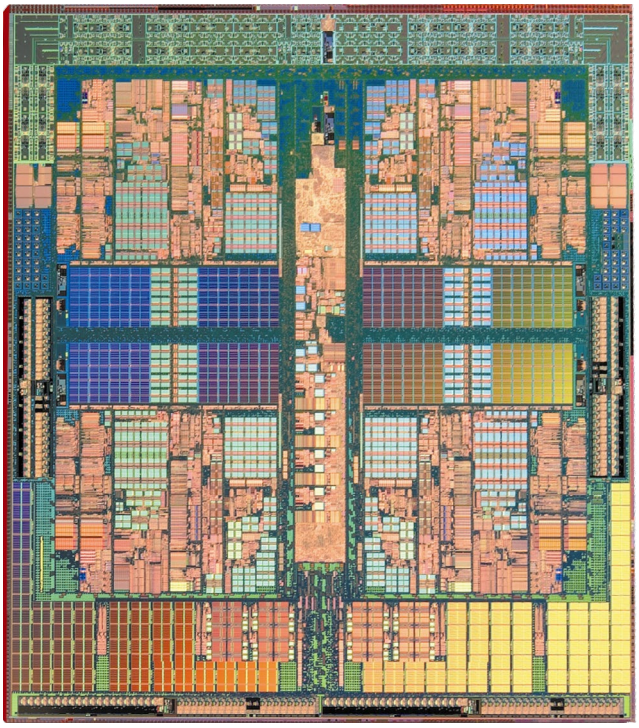
16-bittisten lukujen käsittelyä 8-bittisellä 6502:lla: vasemmanpuoleinen esimerkki lisää muistipaikkoihin $FE ja $FF tallennetun luvun arvoon heksadesimaaliluvun $1234, oikeanpuoleinen kertoo sen kahdeksalla bittisiirtoja käyttäen.

```
lp: cmp r0,r1
    subgt r0,r0,r1
    suble r1,r1,r0
    bne lp
```

Suurimman yhteisen tekijän laskeva silmukka ARMille ehdollista suoritusta käyttäen. Euklideen algoritmi vähentää suuremmasta luvusta pienemmän, kunnes luvut ovat yhtäsuuret.



6502-suorittimen sisukset. Alapuoliskoa hallitsee 8-kaistainen laskenta- ja rekisteriyksikkö, ylimpänä puolestaan näkyy käskyt suoritusvaiheiksi kuvaava mikrokooditaulukko. Näiden välissä on muu toimintalogiikka, esimerkiksi hyppyjen ja lippujen käsittely.



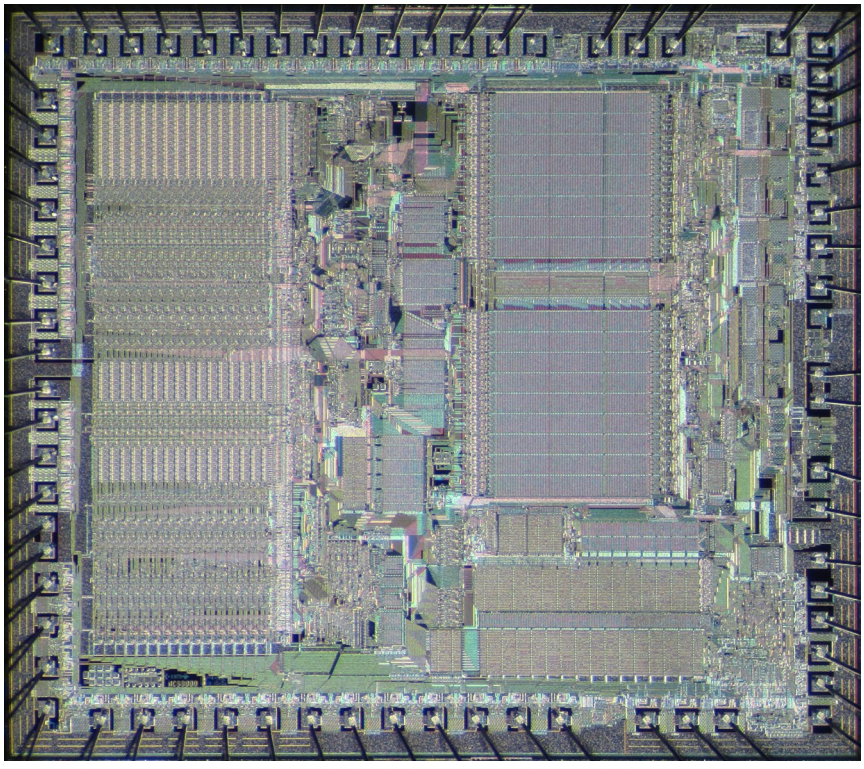
AMD Phenom X4 -suorittimen sisukset. Neljän symmetrisesti asetetun 64-bittisen suoritinytimen alasta valtaosa on välimuistia ja käskyjen purku- ja järjestelylogiikkaa.

rajoitteita: X86:ssa jommankumman operandin on aina oltava rekisteri tai välitön luku, eli esimerkiksi suoraa käskyä ”lisää muistipaikan 1 arvoon muistipaikan 2 arvo” ei ole. Muistiviittaukset voivat kuitenkin cisc-filosofian mukaisesti olla hyvinkin monimutkaisia. Esimerkiksi 32-bittinen X86-käsky `mov eax,[ebx+ecx*4+1256]` muodostaa muistiosoitteen summaamalla vakion ja kaksi rekisteriä, joista ECX:n bittejä vielä siirretään kaksi pykälää vasemmalle.

ARMin kaltaisissa risceissä useimmat käskyt voivat saada operandeikseen vain rekistereitä tai välittömiä lukuja. Muistinkäsittely on hoidettava omilla load-store-käskyillään (`ld`, `st`, `mov`), jotka eivät laske mitään.

ARMin ja 68K:n muistinkäsittelyä tehostavat osoitusmuodot, joissa rekisterin sisältöä kasvatetaan tai vähennetään samalla, kun sitä käytetään muistiosoituksessa. Tämä on kätevää muistialueita läpikäytäessä.

Muistipaikkoihin on monesti kätevää viitata suoran osoitteen sijaan pitämällä käskyn omaa sijaintia kiinnekohtana. 6502:n ja X86:n ehdollisilla hyppykäskyillä voi hypätä enintään 128 tavua eteen- tai taaksepäin, minkä ansiosta käskyn pituudeksi riittää kak-



Motorola 68000:n sisukset. Löydätkö laskenta- ja rekisteriyksikön?

```
lp:  movem (a0)+,(d1-d7)
     movem (d1-d7),-(a1)
     dbne d0,lp
```

```
lp:  subcc r2,r2,#1
     ldmia r0,(r3-r13)
     stmdb r1,(r3-r13)
     bne lp
```

Silmukka, joka kopioi muistialueen sisällön käänteisessä järjestyksessä toiselle muistialueelle rekisterijoukkokäskyjä käyttäen. Vasemmalla 68K, oikealla ARM.

si tavua. Ohjelmakoodia, joka ei käytä suoria osoitteita lainkaan, kutsutaan sijaintiriippumattomaksi (*position-independent*), koska sitä voi ajaa sellaisenaan mistä kohdasta muistia tahansa.

Laskenta on koneen leipätyö

Valtaosa suorittimista laskee oletusarvoisesti binäärimuotoisilla kokonaisluvuilla. 6502:ssa, 68K:ssa ja X86:ssa on tarjolla myös binäärikoodattu desimaali (BCD), jossa kutakin desimaalinumeroa 0-9 vastaa neljä bittiä. Liukulukuja taasen käsitellään omilla liukulukuyksiköillään, joilla on omat rekisterinsä ja käsittelykäskynsä.

Negatiiviset kokonaisluvut ilmaistaan lähes aina niinsanottuna kahden komplementtina, jossa etumerkki vaihdetaan kääntämällä bitit ympäri ja lisäämällä tulokseen yksi. Pelkkiä ykkösiä sisältävä luku on siis arvoltaan -1, aivan kuin kasettinauhurin laskuri, joka 000:sta taaksepäin kelatessa pyörähtää 999:ään. Saman bittijonon voi tulkita joko etumerkillisenä tai etumerkittömänä, ja erot tulevat esiin etenkin kertoessa, jakaessa ja vertaillessa.

Kokonaislukujen yhteen- ja vähennyslaskut (`add`, `sub`) ovat tarjolla joka konekielessä. Kerto- ja jakolaskut (`mul`, `div`) puuttuvat yleensä kasibittisisistä koneista, joissa ne on toteutettava aliohjelmilla tai taulukoilla. Risceissäkin on yleensä suoraan vain kertolasku.

Bittioperaatioita ovat paitsi bitittäiset loogiset operaatiot (`and`, `or`, `eor`/`xor`) ja bittisiirrot (*shifts*), joita on monta lajia. Bittioperaatioiden toiminta on esitetty oheisissa kaavioissa. ”Aritmeettisen” ja ”loogisen” bittisiirron merkitysero on se, että ”aritmeettisessa” luku oletetaan etumerkilliseksi, joten sen ylin bitti pidetään ennallaan.

ARMin erikoisuutena on se, ettei siinä ole lainkaan itsenäisiä käskyjä bittisiirroille, mutta bittisiirron voi yhdistää minkä tahansa laskentaoperaation toiseen lähdeoperandiin. Esimerkiksi `add r0,r1,r2 asr r3` vastaa C-kielen lauseketta `r0=r1+(r2>>r3)`.

Joskus laskutoimituksen tulos ei mahdu kohderekisteriin. Esimerkiksi kahden 8-bittisen luvun summa on 9-bittinen. Summan ylin bitti tallennetaan yleensä niinsanottuun muistinumerolippuun (C, *carry flag*). Muistinumeroa käytetään laskutoimitusten ketjuttamiseen: käskyt `adc`/`addx` ja `sbc`/`sbb`/`subx` ovat yhteen- ja vähennyslaskuja, jotka ottavat mukaan aiemmasta käskystä jääneen muistinumeron.

EX, EXG, XCHG	exchange	Vaihda rekisterien sisällöt keskenään.
LD	load	Lataa muistista.
MOV, MOVE	move	Kopioi dataa rekisteristä tai muistista rekisteriin tai muistiin.
POP, PL	pop, pull	Poimi pinon päällimmäinen arvo.
PUSH, PH	push	Lisää pinon päällimmäiseksi.
ST	store	Tallenna muistiin.

Datansiirto.

ADC, ADDX	add with carry/extend	Summaa muistinumeron kera.
ADD	add	Summaa.
DEC	decrement	Vähennä yhdellä.
DIV	divide	Jaa.
INC	increment	Kasvata yhdellä.
MUL	multiply	Kerro.
NEG	negate	Vaihda etumerkki.
SBB, SBC, SUBX	subtract with borrow/carry/extend	Vähennä muistinumeron kera.
SUB	subtract	Vähennä.

Peruslaskutoimitukset.

AND	and	Bitittäinen ja-operaatio.
ASL, SAL	arithmetic shift left	Siirrä bittejä oikealle jatkaen ylintä bittiä.
ASR, LSR, SHR	[arithmetic/logical] shift right	Siirrä bittejä vasemmalle.
EOR, XOR	exclusive or	Bitittäinen poissulkeva tai.
LSL, SHL	[logical] shift left	Siirrä bittejä oikealle jatkaen nollalla.
NOT	not	Käännä bitit päinvastaisiksi.
OR	or	Bitittäinen tai-operaatio.
ROL, RL, RCL	rotate [with carry] left	Pyöritä bittejä myötäpäivään [C-lipun kautta].
ROR, RR, RCR	rotate [with carry] right	Pyöritä bittejä vastapäivään [C-lipun kautta].

Bittioperaatiot.

Jossittelu

Korkean tason kielten `if`-lausetta vastaa konekielessä useimmiten ehdollinen hyppy. Esimerkiksi käsky nimeltä `beq`, `je` tai `jz` hyppää operandina annettuun muistiosoitteeseen, jos edellisen laskutoimituksen tulos oli nolla. Ennen hyppykäskyä käytetään usein vertailukäskyä `cmp`/`cp`, joka suorittaa vähennyslaskun tallentamatta tulosta mihinkään. Hyppykäskyt on yleensä nimetty vertailun näkökulmasta: jos vähennyslaskun tulos on nolla, niin luvut ovat yhtäsuuret (`e`/`eq`, *equal*).

Tieto laskutoimituksen tuloksesta tallentuu yleensä niinsanottuihin lippuihin, jotka ovat statusrekisterin bittejä. Aiemmin mainittu muistinumerolippu on yksi näistä. Ehdolliset hyppykäskyt tutkivat lippuja ja suorittavat hypyn, mikäli käskyn mukainen ehto täyttyy. Tyypillisiä lippuja ovat:

- Nollalippu (Z, zero), joka ilmaisee, oliko laskun tulos nolla.
- Etumerkkilippu (S, sign tai N, negative), joka vastaa rekisteriin mahtuneen tuloksen ylintä bittiä, joka on negatiivisilla luvuilla 1.
- Muistinumerolippu (C, carry), vastaa laskutoimituksesta yli jäänyttä bittiä.
- Ylivuotolippu (O tai V, overflow) asetetaan, kun muistinumerolippu ei riitä tuloksen jatkeeksi.

6502:ssa, X86:ssa ja 68K:ssa jokainen laskentakäsky vaikuttaa lippuihin. ARMissa lippuvaikutus ilmaistaan käskykohtaisesti käskysanan päätteellä `cc` (*condition codes*). ARMissa ei myöskään aina tarvita ehdollisia hyppyjä, sillä minkä tahansa käskyn suorituksen voi ehdollistaa. Esimerkiksi käsky `addeq` toimii kuten `add`, mutta se suoritetaan vain, jos nollalippu on ylhäällä.

Toisten tavarat pinoon talteen

Tavallinen ehdoton hyppykäsky on nimeltään esimerkiksi `jmp`, `bra` tai `b`, aliohjelmahyppy puolestaan `jsr`, `bsr`, `call` tai `bl`. Aliohjelmakutsut ottavat käskyosoittimen arvon talteen, jotta

BIT, BT, BTST, TEST	bit test	Vertaa yksittäisiä bittejä (ja-operaatio tallentamatta tulosta).
CLf	clear flag	Nollaa lippu (esim. C).
CMP, CP	compare	Vertaa (vähennä tallentamatta tulosta).
Scc, SETcc	set on condition	Aseta rekisterin arvo totuusarvon (esim. NE) mukaiseksi.
SEf, STf	set flag	Aseta lippu (esim. C).

Vertailu ja liput.

Bcc, Jcc	branch/jump on condition	Hyppää, jos ehto (esim. NE) täyttyy.
BL, BAL	branch and link	Hyppää aliohjelmaan, paluuosoite linkkirekisteriin.
DBcc, LOOP	decrement and branch, loop	Vähennä rekisterin arvoa ja hyppää jos ehto täyttyy.
JMP, JP, B, BRA	jump/branch	Hyppää muistipaikkaan.
JSR, JR, BSR	jump/branch to subroutine	Hyppää aliohjelmaan, paluuosoite pinoon.
RET, RTS	return from subroutine	Palaa aliohjelmasta pääohjelmaan.
SWI, INT, TRAP, BRK, SYSCALL	software interrupt, trap, break, system call	Suorita ohjelmallinen keskeytys.

Hyppykäskyt.

HLT	halt	Pysäytä suoritin (jää odottamaan keskeytystä).
NOP	no operation	Älä tee mitään.

Muita käskyjä.

CC, NC	no/clear carry	Muistinumero = 0
CS, C	carry set	Muistinumero = 1
EQ, E, Z	equal/zero	Luvut yhtäsuuret (nollalippu)
GT, G	greater [than]	Ensimmäinen > toinen
LT, L	less [than]	Ensimmäinen < toinen
NE, NZ	not equal/zero	Luvut erisuuret (nollalippu alhaalla)
NS, PL	no sign, plus	Tulos ei-negatiivinen (etumerkki = 0)
S, MI	sign, minus	Tulos negatiivinen (etumerkki = 1)
VC, NO	no/clear overlow	Ylivuotolippu alhaalla.
VS, O	overflow set	Ylivuotolippu ylhäällä.

Ehdot (käskyjen osana).

aliohjelmasta voidaan palata siihen kohtaan, mihin sitä kutsuttaessa jäätiin. Paluukäsky on tyypillisesti nimeltään `ret` tai `rts`.

Vanhemmat käskykannat tallentavat paluuosoitteen yleensä pinoksi kutsutulle muistialueelle. Riscit sen sijaan käyttävät rekisteriä, jonka aliohjelma pinoaa itse mikäli aikoo kutsua muita aliohjelmia. ARMin takaisinlinkittävä hyppykäsky on `bl` (*branch and link*). Linkkirekisteri on yleensä R14 ja käskyosoitin R15, joten aliohjelmasta palataan käskyllä `mov r15,r14`.

Pinoon tallennetaan muutakin kuin paluuosoitteita. Koska aliohjelmat käyttävät samoja rekistereitä kuin pääohjelmakin, on niiden arvoja usein tarpeellista raivata pinoon talteen. Lisäksi pinosta voidaan varata tilaa sellaisille paikallismuuttujille, jotka eivät mahdu rekistereihin. X86:ssa ja 6502:ssa on pino-osoittimeen sidotut push- ja pop/pull-käskyt, kun taas ARMissa ja 68K:ssa pinonkäsittelyyn käytetään tavallisia muistinkäsittelykäskyjä. ARMissa ja 68K:ssa on myös käskyt, joilla haluamansa rekisterijoukon voi tallentaa tai ladata yhdellä kertaa.

Isompien ohjelmakokonaisuuksien kasassa pitämiseksi on yleensä sovittu niinsanottuja kutsukäytäntöjä (*calling convention*). Nämä määrittelevät, miten pääohjelma ja aliohjelma välittävät parametrinsa ja paluuarvonsa, ja mitkä rekisterit aliohjelmalla on lupa sotkea.

Maailma on muistia

Suorittimen näkökulmasta koko ulkoinen maailma on muistia. Muisti jakautuu yleensä 8-bittisen tavun kokoisiin muistialkioihin, joista kullakin on oma numeerinen osoite.

Kun muistiin tallennetaan useista tavuista koostuvia lukuja, on tähän kaksi päätapaa. 68K käyttää laskevaa tavujärjestystä (*big-endian*), eli ensimmäiseen tavuun tulee luvun eniten merkitsevä osa. 6502 ja X86 puolestaan tallentavat alimmat bitit ensin, eli tavujärjestys on nouseva (*little-endian*). ARM toimii kummalla tavujärjestyksellä tahansa, joskin nouseva järjestys on nykyään yleisempi.

Yksinkertaisimmissa laitteissa fyysisillä RAM-, ROM- ja ohjauspiireillä on kiinteät alueet osoiteavaruudessa. VIC-20-ohjelmassa muistipaikkaan `$900F` kirjoittaminen vaikuttaa aina videopiirin värirekisteriin. Monimutkaisemmissa koneissa ohjelmalle näkyvää muistin rakennetta voi sen sijaan muuttaa.

Jos koneessa on enemmän muistia kuin osoiteavaruuteen mahtuu, esimerkiksi 6502-pohjaisessa koneessa yli 64 kilotavua, tarvitaan pankitusta (*banking*). Pankituksessa valitaan, mitkä osat kokonaismuistista näkyvät tietyillä muistiavaruuden alueilla. Nykykäyttöjärjestelmissä muistin näennäistä rakennetta sen sijaan muutellaan, jotta eri prosessit eivät pääsisi käsiksi niille kuulumattomille muistialueille. Samalla koodin oikeutta muuttaa suorittimen tilaa rajoitetaan siirtymällä sen suorittamisen ajaksi niinsanotusta valvojatilasta (*supervisor mode*) käyttäjätilaan (*user mode*).

Näennäismuistin ansiosta kaiken ohjelmalle näkyvän muistin ei tarvitse vastata fyysistä muistia. Jos osoiteavaruus on riittävän suuri, ohjelma voi pyytää käyttöjärjestelmää kuvaamaan vaikkapa koko kiintolevyn sisällön näennäismuistiin. Kun ohjelma yrittää lukea muistipaikkaa, joka ei ole fyysisessä muistissa, se aiheuttaa poikkeustilanteen, jonka käyttöjärjestelmä käsittelee hakemalla levyltä halutun kohdan fyysiseen muistiin. Ohjelman kannalta kone toimii siten, kuin koko levyn sisältö olisi jatkuvasti muistissa.

70-lukulaisilla suorittimilla muistin nopeus ei vielä muodosta pullonkaulaa. Esimerkiksi 6502:lla kannattaa käyttää nopeuskriittisessä koodissa muistinvaraisia taulukoita ja aukirullattuja silmukoita mikäli ne suinkin mahtuvat muistiin. Nykysuorittimilla laskuoperaatio saa sen sijaan olla jo todella monimutkainen, jotta sille kannattaisi laskea valmis taulukko. Aukirullauskaan ei enää kannata: sisäisten välimuistien ja älykkäiden liukuhihnojen ansiosta se todennäköisemmin hidastaa kuin nopeuttaa koodia.

```
        !to "skrolli.prg",cbm
        *=$0801     ; Ohjelman alkuosoite.

; Pakollinen BASIC-osuus 10 SYS2061 + loppunollat:
!byte $0b,$08,$0a,$00,$9e,$32,$30,$36,$31,0,0,0

        ldx #0      ; Laskuri (X) nollaan.

silmu   txa         ; X akkuun jotta saadaan
        and #15     ; laskettua X AND 15.
        tay         ; Tulos Y:hyn ja haetaan
        lda herja,y ; tavu osoitteesta herja+Y.

        sta $0400,x ; Kirjoitetaan se
        sta $0500,x ; näyttömuistin kullekin
        sta $0600,x ; 256-tavuiselle
        sta $0700,x ; lohkolle kohtaan X.

        inx         ; Kasvatetaan X:ää.
        bne silmu   ; Toistetaan kunnes pyörähti.

        rts         ; Palataan BASIC-tulkkiin.

herja   !scr "tilaa skrolli!! "
```

6502-esimerkki Commodore 64:lle. ACME-ristiinkääntäjän tuottama PRG-tiedosto käynnistyy suoraan vaikkapa VICE-emulaattorissa.

```
        bits 16         ; Nasm 16-bittiseen tilaan.
        org 0x100       ; COM-ohjelman alkuosoite.

        mov ax,0xb800   ; Näyttömuistin alkuosoite
        mov es,ax       ; .. segmenttirekisteriin.
        xor di,di       ; Kohdeosoitin nollaksi.

        mov ah,14       ; Väri AX:n ylempään tavuun.

silmu1  mov si,herja    ; Lähdeosoitin tekstin alkuun.
        mov cx,16       ; Silmukkalaskuriksi 16.

silmu0  lodsb           ; AL <- [DS*16+SI], SI kasvaa.
        stosw           ; AH*256+AL -> [ES*16+DI], DI +2.
        loop silmu0     ; CX vähenee, toistetaan kunnes 0.

        cmp di,80*25*2  ; Ollaanko käyty koko näyttö?
        jne silmu1      ; Jos ei, jatketaan toistoa.

        ret             ; Palataan komentotulkkiin.

herja   db "Tilaa Skrolli!! "
```

16-bittinen X86-esimerkki MS-DOSille. Ohjelma kääntyy NASMilla suoraan COM-tyyppiseksi ohjelmatiedostoksi.

Laitteita ohjaamaan

Tietokonelaitteistoon kuuluu oheispiirejä, joilla on omat ohjausrekisterinsä. 6502:ssa, 68K:ssa ja ARMissa nämä rekisterit näkyvät osana muistiavaruutta. X86 taasen käyttää tähän erillisiä I/O-portteja, joita käsitellään `in`- ja `out` -käskyillä.

Jotta suorittimen ei tarvitsisi jatkuvasti kysellä eli pollata laitteiden tiloja, on olemassa keskeytykset: laite voi lähettää suorittimelle keskeytyskutsun (IRQ, *interrupt request*), joka saa sen keskeyttämään senhetkisen toimintansa ja siirtymään käsittelyrutiiniin. Useimmat käyttöjärjestelmät suorittavat joitakin kymmeniä kertoja sekunnissa ajastinkeskeytyksen, jotta tietyt juoksevat asiat saadaan hoidettua.

Yksinkertaisimmillaan keskeytys ei juuri eroa aliohjelmakutsusta. Aliohjelman alkuosoite haetaan keskeytyksen tyypin ja mahdollisen numeron mukaan hyppytaulukosta. Nykykäyttöjärjestelmissä keskeytys myös siirtää suorittimen valvojatijaan. Vain valvojatilassa toimiva käyttöjärjestelmä voi käsitellä ulkoisia laitteita, ja sovellukset tekevät käyttöjärjestelmän apua tarvitessaan niinsanotun ohjelmallisen keskeytyksen (NMI, *non-maskable interrupt*).

Monta käskyä samaan aikaan

Yleisesti käytetyt käskykannat periytyvät vuosikymmenten takaa, mutta suorittimien toimintatavat ovat muuttuneet tänä aikana huomattavasti. Erityisesti rinnakkaisuutta on tullut lisää.

Perinteiset cisc-suorittimet ajavat vain yhtä käskyä kerrallaan. Käskyn suoritus jakautuu useisiin peräkkäisiin vaiheisiin, jotka on koodattu suorittimen sisäiseen mikrokooditaulukkoon. 6502-käskyn suoritus koostuu 2–8 vaiheesta, kun taas 8086:n jakolasku vie yli sata kellojaksoa. Ohjelmoija voi laskea koodinsa suoritusajan tällaisella suorittimella yksinkertaisesti summaamalla käskyjen viemät kellojaksot ja jakamalla tuloksen kellotaajuudella.

Riscin kantaviin ajatuksiin kuuluu, että yksinkertaisten käskyjen suoritusvaiheet voivat olla rinnakkaisia. Alkuperäisessä Archimedeen ARM-suorittimessa on kolmivaiheinen liukuhihna: suoritin tallentaa yhden laskutoimituksen tuloksen rekisteriin samaan aikaan, kun se laskee seuraavan käskyn laskutoimituksen ja lukee muistista sitä seuraavan käskyn.

Liukuhihnatekniikka tekee hypyistä suhteessa kalliita. Kun hyppykäsky toteutuu, joudutaan sen jälkeen tulevien käskyjen suoritusvaiheet hylkäämään. Ongelman ehkäisemiseen on useita tapoja. ARMin ehdollinen käskynsuoritus on yksi näistä: yhden tai parin käskyn suorittamatta jättäminen on edullisempaa kuin liukuhihnan tyhjennys. Edistyneempi tekniikka on haarautumisen ennakointi (*branch prediction*), jossa suoritin arvaa etukäteen, tuleeko hyppy toteutumaan, ja lataa käskyjä liukuhihnalle sen mukaan. Spekulatiivisessa suorituksessa puolestaan suoritetaan molemmat vaihtoehdot ja jätetään vain toisen vaikutukset voimaan.

Monissa suorittimissa on useita rinnakkaisia liukuhihnoja, eli ne suorittavat peräkkäisiä käskyjä aidosti samaan aikaan. Usein peräkkäiset käskyt ovat kuitenkin riippuvaisia toistensa tuloksista, joten ohjelmoijan tai suorittimen on hyvä järjestää käskyt siten, etteivät peräkkäiset käskyt käytä samoja rekistereitä. Suorittimen automatiikassa tällaisia tekniikoita ovat *out-of-order execution* ja *register renaming*.

X86 on tuottanut aivan omia haasteitaan suoritinsuunnittelijoille. 90-luvun puolestavälistä alkaen monimutkaiset X86-käskyt on yleensä pilkottu risc-tyylisiksi mikrokäskyiksi, joihin sitten sovelletaan edellämainittuja tekniikoita.

Erikoiskäskyjä erikoistehtäviin

Vaikka peruskäskykannoilla pystyykin jotenkuten tekemään kaiken, on niille

```
# Määritellään linkkeriä varten symboli _start,
# joka osoittaa ohjelman ajon alkukohtaan.

.globl _start
_start:

# Alustetaan silmukkalaskuri.

        movq $1024,%rbp     mov r8,#1024

# Suoritetaan järjestelmäkutsu write(1,herja,15),
# missä 1 on standardiulostulo ja 15 pituus.
# Write-kutsun numero on 64-bittisessä Linuxissa 1
# ja 32-bittisessä 4.

silmu:  movq $1,%rax        mov r7,#4
        movq %rax,%rdi      mov r0,#1
        movq $herja,%rsi    adr r1,herja
        movq $15,%rdx       mov r2,#15
        syscall             swi 0

# Vähennetään laskuria, hypätään jos ei nolla.

        decq %rbp           subcc r8,r8,#1
        jnz silmu           bne silmu

# Suoritetaan järjestelmäkutsu exit(0)

        movq $4,%rax        mov r7,#1
        xorq %rdi,%rdi      mov r0,#0
        syscall             swi 0

# Tulostettava merkkijono:

herja:  .string "Tilaa Skrolli! "
```

Kernel-kutsuja käyttävä Linux-esimerkki 64-bittiselle X86:lle (vasemmanpuoleiset käskyt) ja 32-bittiselle ARMille (oikeanpuoleiset). Ohjelma kääntyy kohdekoneella käskyllä gcc -nostdlib ohjelma.s -o ohjelma tai erikseen as-assembleria ja ld-linkkeriä kutsumalla.

monesti olemassa erityisiä laajennoksia, jotka nopeuttavat tietyntyyppisten tehtävien suorittamista.

Liukulukumatematiikkaa on vanhastaan tarvittu etenkin tieteellisessä laskennassa. Ideana on, että luvut esitetään käyttäen kantalukua ja ”nollamäärää” (eli mantissaa ja eksponenttia), jolloin niiden arvoalue on huomattavasti laajempi kuin kokonaisluvuilla. PC-suorittimiin alkoi tulla integroituja liukulukuyksiköitä 486-aikakaudella, mutta pelikonsoli- ja mobiilimaailmassa liukulukulaitteisto ei ole ollut vielä 2000-luvullakaan mikään itsestäänselvyys.

Kuva- ja äänidatan käsittelyn nopeuttamiseen on käytetty digitaalisia signaaliprosessoreita (DSP). 90-luvulla DSP-tyyppisiä SIMD-käskystöjä (*single instruction, multiple data*) alkoi ilmestyä myös perussuorittimiin. Esimerkkejä SIMD-laajennoksista ovat X86:n MMX ja SSE sekä ARMin NEON.

SIMD-käskyt operoivat nimensä mukaisesti usealla erillisellä data-alkiolla rinnakkain. Esimerkiksi MMX-käsky `paddb mm0,mm1` tulkitsee 64-bittisten multimediarekisterien MM0 ja MM1 arvot erillisten 8-bittisten tavujen riveinä laskiessaan ne yhteen. Käskyjä on myös esimerkiksi data-alkioiden uudelleenjärjestelyyn.

SSE:n ja NEONin rekisterit ovat 128-bittiset, ja alkiot voivat olla myös liukulukuja. SSE tukee liukuluvuille myös neliöjuuren kaltaisia monimutkaisia toimituksia, ja nyky-X86:ssa se onkin syrjäyttänyt vanhat X87-käskyt.

Etenkin mobiilimaailmassa yksiin kuoriin on integroitu valtava määrä erikoistunutta laskentalogiikkaa. Esimerkiksi Qualcommin Snapdragon 810 -piiri sisältää kahdeksan 64-bittistä ARM-suoritinydintä, joista kullakin on kolme erillistä liukuhihnaa ja NEON- ja liukulukulaajennokset. Lisäksi piirillä on 288-laskentayksikköinen grafiikkasuoritin, 32-bittinen DSP sekä eri radioprotokolliin erikoistuneita ohjainpiirejä. Samanaikaisia laskutoimituksia saattaa siis tapahtua taskussa enemmän kuin entisajan supertietokoneessa.

Ja sitten räpeltämään

Luontevimmat ja usein palkitsevimmat kohteet konekielitason askarteluun löytyvät yksinkertaisesta tietotekniikasta – esimerkiksi vanhoista kotimikroista, sulautetuista järjestelmistä ja Arduinon kaltaisista elektroniikka-alustoista. Laitteen toimintaan on näissä mahdollista sukeltaa yksittäisten bitinliikkeiden ja kellojaksojen tarkkuudella, ja suorittimen erityispiirteitä voi hyödyntää tavoilla, jotka katoavat korkean tason kielillä näkymättömiin. Pieniä laitteita konekieliohjelmoidaan useimmiten ristiinkääntäjillä (*cross-assembler*), ja apuna voi usein käyttää myös emulaattoreita. Valmiita oppaita kannattaa etsiä kohdekoneen mukaan.

Isommilla laitteilla helpoin tie konekielen pariin löytyy korkean tason kielten kääntäjistä: esimerkiksi GCC kääntää `-S`-vivulla lähdekooditiedoston assemblyksi, jota voi tutkia ja muokata itse. Kääntäjät tukevat myös inline-assemblyä, eli assembly-osien upottamista korkean tason kielen sekaan. Koodin nopeuttamista ei kannattane käyttää motivaattorina nykysuorittimien konekieliohjelmoinnin opetteluun – sen sijaan kannattaa kokeilla esimerkiksi mahdollisimman pienten ohjelmien kirjoittamista.

Assembly-kääntäjien ohella voi käyttää myös suorempia työkaluja. Konekielimonitorit ja debuggerit on tarkoitettu muistin ja muistinvaraisen ohjelman käsittelyyn lennossa. Heksaeditoreilla voi puolestaan tutkia ja muokata ohjelmatiedostoja suoraan, ja monet niistä osaavat esittää tiedoston sisällön myös assemblynä.

Tämä artikkeli oli tiivis pintakatsaus konekielen olemukseen. Sen tiedoilla voi yrittää tutkia assembly-koodia, mutta syvempään tutustumiseen on hyvä olla kattavat käskykanta- tai suoritinkohtaiset dokumentit. Tämän jälkeen konekieliohjelmoinnin saloihin pääseekin parhaiten sisään valitsemalla itselleen jonkin mieluisen projektin ja ohjelmoimalla itse.

Aikavälikertaus muistin tukena

Harmittaako unohtelu? Mitä jos siirtäisit vastuun muistamisestasi algoritmille? Aikavälikertausohjelmat hoitavat homman.

Teksti: Jussi Määttä Kuvat: Sakari Leppä, Jussi Määttä, Mitol Meerna

Jokainen meistä on joskus opetellut asioita ulkoa. Se on kovaa työtä, ja ahkerankin pänttäämisen tuloksilla on taipumus haihtua muistista nopeasti. Jos muistiin taotut tiedonmuruset ovat tärkeitä tai hyödyllisiä, niiden unohtaminen voi olla valtava vahinko.

Unohtamista voi tietysti ehkäistä kertaamalla. Kaikkea oppimaansa ei tosin voi kerrata päivittäin, eikä siinä mitään järkeä olisikaan. Nimittäin mitä paremmin jokin fakta on muistiin iskostunut, sitä kauemmin kestää ennen kuin se unohtuu. Moni muistaa yhä puhelinnumeroita ajalta ennen kännyköitä, ja niitä nyt muistelemalla muistijäljet vahvistuvat entisestään. Hyvin opittua ei tarvitse kerrata usein; vastikään opittu taas vaatii enemmän työtä säilyäkseen muistissa.

Tietokoneet apuun

On olemassa tietokoneohjelmia, jotka pyrkivät ajoittamaan kertaamisen mahdollisimman hyvin. Käyttäjä syöttää ohjelmaan kysymys–vastaus-pareja, joita ohjelma sitten kuulustelee. Jos käyttäjä muistaa oikean vastauksen vaikkapa kaksi päivää kysymyksen lisäämisen jälkeen, ohjelma saattaa ajoittaa seuraavan kertauksen neljän päivän päähän. Jos oikea vastaus löytyy silloinkin, voi seuraava *intervalli* eli aikaväli olla kahdeksan päivää. Ihmisen muisti toimii siten, että kertausten karttuessa sopivat intervallit kasvavat kuukausien tai jopa vuosien mittaisiksi.

Tällaisia ohjelmia kutsutaan *aikavälikertausohjelmiksi* (engl. spaced repetition software, SRS). Niiden käyttämät aikavälien laskentakaavat pohjaavat tieteelliseen tietoon muistin toiminnasta. Kyse ei sentään ole The Matrix -elokuvassa nähdyn kaltaisesta ”uploadauksesta”, mutta mielleyhtymiltä ei voi välttyä, kun teknologia auttaa optimoimaan ihmisen muistia.

Täytyy korostaa, että aikavälikertaus on enemmän kuin vain joukko kysymyksiä ja vastauksia, joita kuulustellaan pahvikorttien tai tietokoneohjelman avulla. Aikavälikertauksessa tietokone arvioi, kuinka pitkään käyttäjä voi huoletta olla kertaamatta tietoa. Nyrkkisääntönä on, että kannattaa kerrata juuri ennen kuin tieto olisi unohtunut.

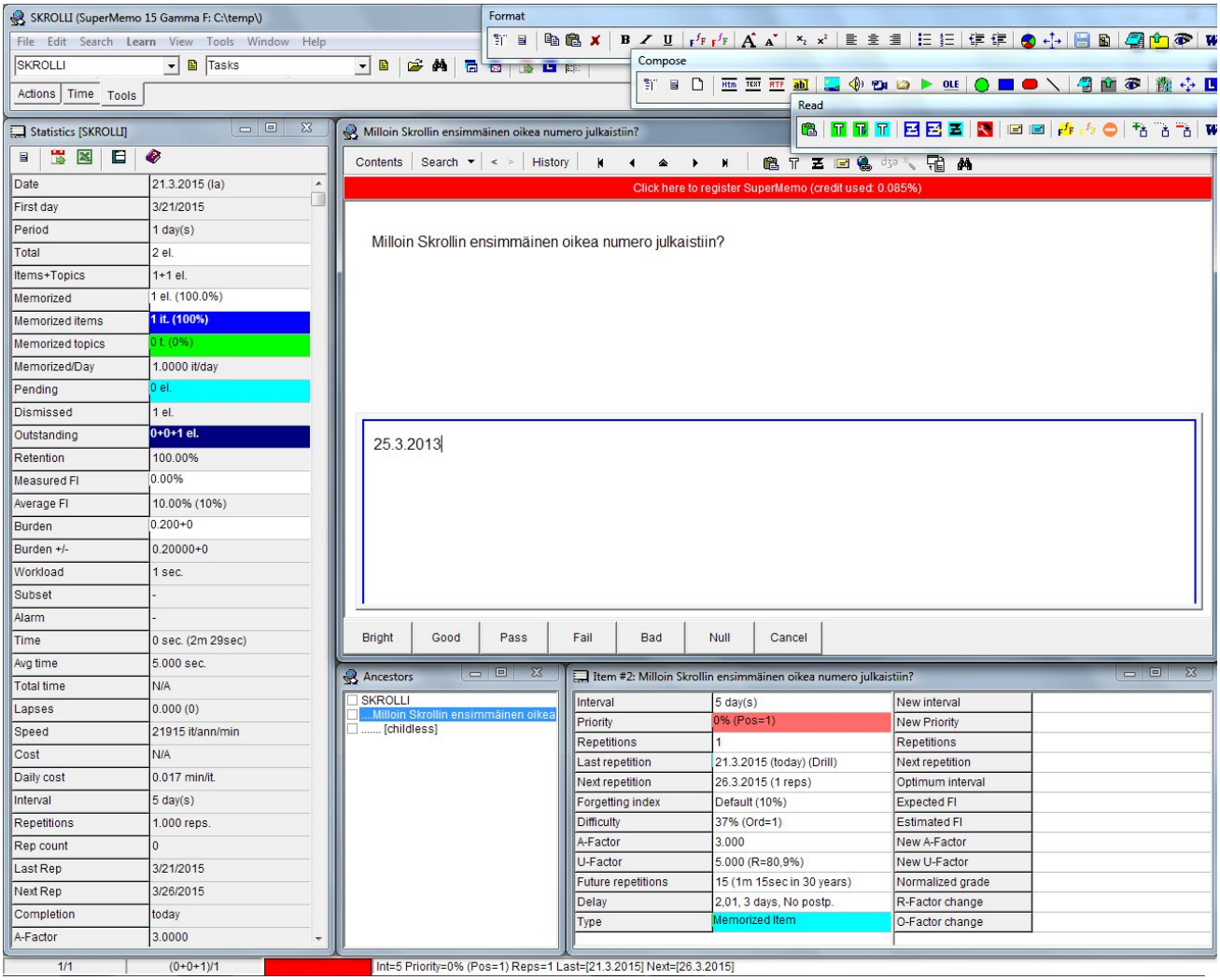
Unohtamisen lait

Saksalainen psykologi Hermann Ebbinghaus on tärkeä mies muistamisen ja unohtamisen tutkimuksen historiassa. Vuonna 1885 julkaisemassaan tutkimuksessa hän kuvasi, kuinka hän oli opetellut ulkoa satunnaisia kirjainyhdistelmiä ja testannut, miten pitkään

Yleistietoa muistitekniikoista

Muisti ja keinot sen vahvistamiseksi eivät lakanneet kiinnostamasta ihmisiä kirjoitustaidon keksimisen myötä. Muinaiset kreikkalaiset ja roomalaiset käyttivät monenlaisia muistisääntöjä ja -tekniikoita esimerkiksi puheiden ulkoa opetteluun. Keskiajalla muistitekniikoihin yhdistettiin myös erilaisia okkultistisia järjestelmiä.

Mainio katsaus aiheeseen on F. A. Yatesin kirjassa **The Art of Memory** (1966). Kiinnostavalta vaikuttaa myös tuoreempi J. Foerin **Moonwalking with Einstein: The Art and Science of Remembering Everything** (2011).



Supermemon tehokäyttäjille suunnattu Warrior-tila ei sovellu hellämielisille.

ne säilyivät muistissa.

Tutkimuksensa tuloksena Ebbinghaus sai tukea hypoteesilleen unohduskäyristä (engl. forgetting curve). Kyseessä on verrattain yksinkertainen matemaattinen malli, joka kuvaa, kuinka muistijälki alkaa heiketä kertauksen jälkeen. Heikkenemisvauhti riippuu siitä, miten vahvasti tieto on muistiin kirjoittunut, ja tätä voidaan arvioida laskemalla tehtyjen kertausten lukumäärä.

Suunnilleen sata vuotta Ebbinghausin tutkimuksen julkaisun jälkeen, vuonna 1987, puolalainen Piotr Woźniak julkaisi ensimmäisen version Supermemo-ohjelmastaan. Sen 1.0-versio on tiettävästi ensimmäinen julkaistu aikavälikertausohjelma. Woźniak oli kiinnostunut muistista jo vuosia aiemmin yliopisto-opintojensa innoittamana, ja Supermemo oli pitkällisen suunnittelun ja kokeilun tulos. Ennen ohjelman toteuttamista Woźniak testasi algoritminsa toimivuutta parin vuoden ajan laskemalla kertausaikavälejään kynän ja paperin avulla.

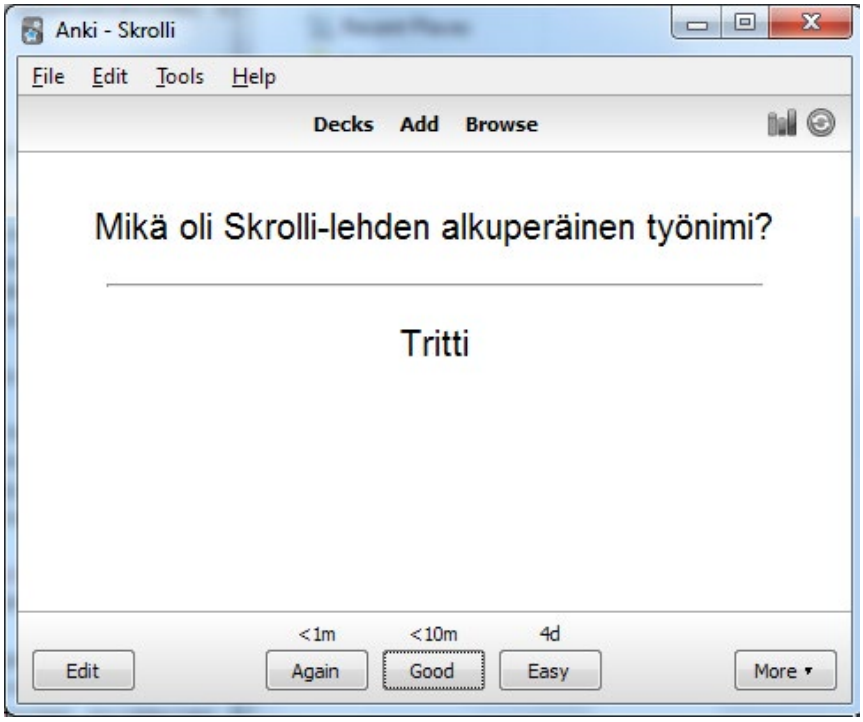
Pohjimmiltaan aikavälikertausohjelmat siis yrittävät arvioida, kuinka tukevasti mikäkin fakta on käyttäjän mielessä, ja laskea sitten unohduskäyrän avulla, milloin asia kannattaa seuraavan kerran kerrata. Käytännössä tämä tapahtuu siten, että kun ohjelma esittää käyttäjälle kysymyksen, käyttäjä vastaa siihen mielessään ja painaa sitten nappia, jolloin ohjelma näyttää oikean vastauksen. Ohjelma pyytää käyttäjää kertomaan, muistiko tämä vastauksen vai ei, oliko kysymys helppo vai muistaako käyttäjä koskaan edes nähneensä vastausta. Tämän perusteella algoritmi päivittää arvionsa muistijäljen vahvuudesta ja laskee uuden intervallin.

Käyttökohteita ja -tapoja

Tyyppiesimerkki aikavälikertauksen käytöstä on vieraan kielen sanaston opettelu. Yleisesti pidetään hyvänä, että kysymys–vastaus-parit ovat mahdollisimman yksinkertaisia ja nopeasti käsiteltäviä. Sanaston pänttäyksessä tulisi siis välttää yhdistämästä useita eri sanoja samaan kysymykseen. Woźniak kutsuu tätä pienimmän informaatiomäärän periaatteeksi.

Kysymyskortteja laatiessa on syytä kiinnittää huomiota myös aktiivisen ja passiivisen muistamisen eroon. Suomennoksen antaminen vieraan kielen sanalle on passiivista muistamista ja suomenkielisen sanan kääntäminen vieraalle kielelle aktiivista. Kielen ymmärtäminen on passiivinen taito, kun taas sen tuottaminen on aktiivinen taito. Oppimisen kannalta lienee parasta tehdä jokaisesta sanasta kortti kumminkin päin.

Kieltä ei opi pelkästään sanastoa tankkaamalla, mutta aikavälikertaus taipuu myös luetun ymmärtämisen harjoitteluun. Monet ovat käyttäneet varsinkin japanin opiskeluun niin kutsuttua AJATT-metodia, jonka nimi-



Avoimen lähdekoodin Anki on aikavälikertausohjelmien parhaimmistoa.

Eräitä kertausohjelmia

SRS-sovelluksista parhaimpia lienevät avoimen lähdekoodin **Anki** (http://ankisrs.net/) ja **Mnemosyne** (http://mnemosyne-proj.org/). Molemmat ovat osoittautuneet luotettaviksi pitkälläkin aikavälillä, ja niillä on laajat käyttäjäkunnat. Anki mahdollistaa korttien helpon synkronoinnin ohjelman pääkehittäjän ylläpitämälle palvelimelle, ja kertauksia voi tehdä myös ilmaisella Android-sovelluksella tai maksullisella Iphone-sovelluksella. Toisaalta yksinkertaisemmalla Mnemosynellä on helpompi päästä alkuun.

Memrise (http://memrise.com/) ja **Cerego** (http://cerego.com/) toimivat nettiselaimessa ja älypuhelimessa ja vaikuttavat helppokäyttöisiltä. Niihin on tarjolla paljon valmiita kysymyskokoelmia, joiden avulla aikavälikertausta voi kokeilla vaivattomasti.

Klassinen **Supermemo** (http://supermemo.com/), josta kaikki alkoi, on myös voimissaan. Windows-pohjainen ohjelma on ehtinyt jo versioon 16. Sitä ei voi suositella muille kuin asiaan todella vihkiytyneille, sillä ohjelman lukemattomat toiminnot ja äärimmäisen sekava käyttöliittymä vaativat viikkojen opettelun. On myös Supermemo-nimeä kantavia älypuhelinsovelluksia, mutta ne ovat lähinnä Puolan sisämarkkinoille suunnattuja.

merkki Khatzumoto esitteli blogissaan viime vuosikymmenellä. Siinä tärkeässä osassa on aikavälikertauksen ahkera käyttö siten, että kunkin kortin sisältönä on kysymyspuolella kokonainen vieraskielinen lause ja vastauspuolella sanojen käännöksiä tai muita ymmärryksen apuja.

Aikavälikertaus sopii myös trivian opiskeluun. Yhdysvalloissa suositussa Jeopardy-televisiovisailussa ennätyksellisen hyvin menestynyt Roger Craig käytti avoimen lähdekoodin kertausohjelma Ankia ahtaessaan nippelitietoa päähänsä. Toivoa sopii, että Craig lahjoitti satojen tuhansien dollareiden palkintorahoistaan muutaman hilkun Ankin kehittäjälle!

Muitakin hyödyllisiä tai viihteellisiä käyttökohteita on helppo keksiä. Kertausohjelmaan voi syöttää vaikkapa satoja Emacs-tekstieditorin näppäinkomentoyhdistelmiä, kuvat maailman valtioiden lipuista tai kaikki vastaukset työpaikan kahvihuoneen ainaiseen ihmetyksen aiheeseen siitä, miksi milloinkin on liputuspäivä.

Aikavälikertauksen ongelmat

Aikavälikertaus on saanut osakseen arvostelua yhtä kauan kuin ohjelmia on ollut tarjolla. Osa argumenteista on tyhjänpäiväisiä (esim. ”Miksi opetella ulkoa? Ainahan voi katsoa netistä.”), mutta aiheellistakin kritiikkiä riittää.

Jos jonakin päivänä jättää tekemättä ohjelman ajoittamat kertaukset, ne lisäävät seuraavan päivän kertausmäärää. Tekemättömillä kertauksilla on tapana kasaantua esimerkiksi lomien aikana, ja ohjelman antama tyly ”1000 korttia odottaa kertausta” -ilmoitus syö helposti innostuksen koko asiaan. Jokainen aikavälikertausta käyttämään ryhtynyt huomaa pian, että selkeän rutiinin muodostaminen on tärkeää.

Ohjelmien kehittäjät ovat yrittäneet keksiä kertausten kasaantumisen pulmaan erilaisia ratkaisuja, mutta pohjimmiltaan ongelmaa voi vain piilotella tai lykätä. Ainoa todellinen ratkaisu on ottaa itseään niskasta kiinni ja tehdä kertaukset.

Aikavälikertaus voi myös aiheuttaa ahdistusta. Jos on käyttänyt paljon aikaa kysymysten ja vastausten syöttämiseen, voi kertausten tekemättä jättäminen tuntua aiemman työn hukkaan heittämiseltä. Ja jos kertauksista pitää taukoa, voi ajatus kertyvästä korttivuoresta tuntua stressaavalta.

Toisaalta kertausohjelmista voi innostua liikaa. Jotkut päättävät syöttää ohjelmaan kaiken, mitä he kuvittelevat haluavansa muistaa. Jos huomaa miettivänsä, pitäisikö ohjelmaan syöttää kummitädin syntymäpäivä tai Norjan lipun mittasuhteet, on syytä pohtia, onko harrastus lähtemässä käsistä.

Jotkut kertausohjelmien käyttäjät alkavat pitää ohjelmien algoritmeja kaikkivoipina oppimestareina, joita ei sovi missään tapauksessa uhmata. Netin keskustelupalstoilla voi tavata huolestuneita käyttäjiä, jotka ovat törmänneet vieraan kielen sanaan kertausohjelman ulkopuolella ja pelkäävät algoritmin sekoavan – aivan kuin he opiskelisivat kieliä algoritmia eivätkä itseään varten!

Kannattaako?

Onko aikavälikertaus todella vaivansa arvoista? Nimimerkki Gwern laskeskelee *Spaced Repetition* -artikkelissaan (ks. linkki jutun lopussa), että jos jonkin faktan etsimiseen tulisi muuten käyttämään elämänsä aikana yhteensä vähintään viisi minuuttia, niin sen lisääminen kertausohjelmaan säästää aikaa. Asia ei kuitenkaan ole aivan näin yksinkertainen. Tiedon tarpeellisuutta

on joskus vaikea arvioida etukäteen, mutta kertausohjelman näkökulmasta jokainen siihen lisätty fakta on samanarvoinen. Supermemossa voi tosin säätää prioriteetteja, mutta toiminto vaatii käyttäjältä lisätyötä, ja sen hyöty on kyseenalainen.

Reaalimaailma sen sijaan tekee automaattista priorisointia: hyödyllisiä tietoja tarvitaan useammin, joten niitä tulee kerranneeksi luonnostaan silloin tällöin. On myös syytä huomata, että vaikkapa harvinaista vieraan kielen sanaa ei välttämättä tarvitse koskaan erikseen opetella, vaan merkityksen voi päätellä asiayhteydestä. Toisaalta elämässä pärjää hyvin, vaikka ei tuntisikaan jokaista obskuuria sivistyssanaa.

Voidaan myös kysyä, onko kaikki tieto todella redusoitavissa yksittäisiksi kysymys–vastaus-pareiksi. Miten vaikkapa filosofinen essee muokkautuisi aikavälikertaukseen soveltuvaksi? Kysymys on kiistanalainen. Yhdeksi ratkaisuksi on ehdotettu ns. cloze-tyyppisiä kortteja, joissa kysymykset koostuvat lauseista, joista on peitetty jokin tärkeä sana.

Aikavälikertaukseen soveltuukin parhaiten materiaali, joka on luonnollisesti jaettavissa pieniin ja mahdollisimman itsenäisiin yksiköihin. Opiskeltavan aihepiirin on myös hyvä olla selkeästi rajattu, jotta ohjelmaan tulee syöttäneeksi ainoastaan hyödyllistä ja ulkoa muistamisen arvoista informaatiota.

Materiaalin opiskeluun tulisi varata riittävästi aikaa, vähintään kuukausia, sillä aikavälikertauksen hyöty korostuu nimenomaan pitkällä aikavälillä. Toisaalta väitän, että ideaalitilanteessa ohjelmaan syötetty materiaali muuttuu vähitellen käyttäjälle turhaksi. Tällöin ohjelma täyttää tarkoituksensa oppimisen tehostajana mutta ei saa käyttäjää tuntemaan, että hänen muistinsa on algoritmin armoilla.

Esimerkiksi kieltenopiskelija voi taitojen ja sanavaraston karttuessa siirtyä käyttämään kieltä päivittäisessä elämässään, jolloin hän pääsee luonnostaan edellä mainitun ”reaalimaailman priorisoinnin” piiriin. Monet ammatilliset tietokokonaisuudet taas vanhentuvat vähitellen: vaikkapa IT-alalla päntätään usein sertifiointitutkintoihin, jotka voivat menettää merkityksensä muutamassa vuodessa uusiin järjestelmiin siirryttäessä.

Mikäli materiaali on selkeästi rajattu, jaettavissa pieniin yksiköihin ja tulee aikanaan vanhentumaan tai muuttumaan osaksi käyttäjän arkea, kannattaa harkita vakavasti aikavälikertauksen käyttöä. Kaupallista SRS-alustaa pyörittävä Cerego-yhtiö on julkaissut useita tutkimuksia, joissa pyritään osoittamaan empiirisesti oppimisen tehostumista. Näiden tutkimusten objektiivisuutta en lähde arvioimaan. Internetistä on myös helppo löytää tavallisten ihmisten tarinoita positiivisista oppimiskokemuksista.

Lopullinen vastuu oppimisesta ja sen järkevyydestä säilyy aina käyttäjällä itsellään. Kieliä opeteltaessa on muistettava, että puhumista ja omaehtoista lauseiden tuottamista ei opita kertausohjelmilla vaan tekemällä. Sama pätee useimpiin muihinkin taitoihin: ei riitä, että muistaa kaiken, jos tietoa ei osaa hyödyntää.

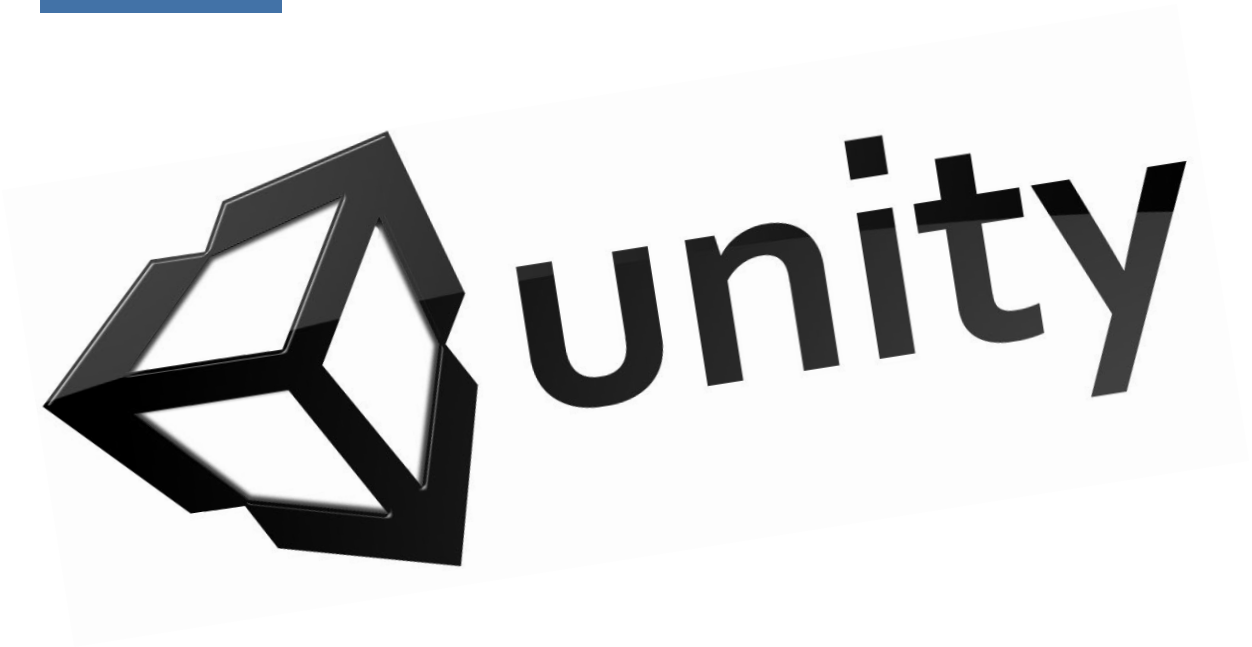
Lopuksi

Vaikka ei juuri nyt olisikaan tarvetta optimoida oppimistaan tai oppia mitään uutta, on aikavälikertauksen olemassaolosta hyvä tietää. Tietotekniikka mahdollistaa kyllä verrattain nopean tiedonhaun, mutta muistilla on yhä arvonsa. Aiheeseen tutustuminen lisää myös ymmärrystä omista oppimis- ja muistamisprosesseista. Ja onhan ajatus kognitiivisten toimintojen tehostamisesta teknologian avulla jännittävä jo itsessään!

Tolkku on kuitenkin pidettävä mukana. Stoalaisfilosofi Seneca kirjoitti: ”Pidätkö moitittavana henkilöä, joka katsoo turhat asiat hyödyllisten veroisiksi ja levittelee talossaan nähtäville kallisarvoisia esineitä, mutta etkö pidä moitittavana sitä, joka on täyttänyt mielensä tiedollisella rihkamalla?”

Lähteitä ja lisämateriaalia

- History of SuperMemo. http://supermemo.com/english/history.htm [Aikavälikertauksen ja Supermemon historiaa.]
- Ebbinghaus, H. (1913): Memory: A Contribution to Experimental Psychology. Engl. käänt. H. A. Ruger ja C. E. Bussenius v. 1885 saksaksi julkaistusta alkuteoksesta. http://psychclassics.yorku.ca/Ebbinghaus/index.htm [Ebbinghausin klassinen artikkeli koekuvauksineen.]
- Wolf, G. (2008): Want to Remember Everything You'll Ever Learn? Surrender to This Algorithm. Wired Magazine 16.05. http://archive.wired.com/medtech/health/magazine/16-05/ff_wozniak?currentPage=all [Mainio artikkeli Supermemosta ja sen omalaatuisesta isästä P. Woźniakista.]
- Gwern: Spaced Repetition. http://www.gwern.net/Spaced%20repetition [Yhteenveto aikavälikertauksesta. Sisältää laajan kirjallisuuskatsauksen.]
- AndrewFMs. (2011): Simulation of SRS vs. Traditional Review. https://www.youtube.com/watch?v=ai2K3qHpC7c [Video havainnollistaa erilaisia kertausstrategioita.]
- Baker, S. (2011): Final Jeopardy. Mariner Books. [Luvussa 10 kerrotaan, kuinka Roger Craig käytti Anki-ohjelmaa valmistautuessaan tietokilpailuun.]
- Kenny, P.: A Warning About Flashcard and Spaced Repetition Software. http://nihongoperapera.com/flashcards-insufficient.html [Aikavälikertauksen hyödyistä ja hyödyttömyydestä kieltenopiskelussa.]
- Wyner, G. (2014): Fluent Forever. Harmony Books. [Mainio opas tulokselliseen kielten itseopiskeluun. Neuvoo myös aikavälikertauksen käytössä.]



Unity 3D – pelinkehityksen vallankumous

Viime vuosien aikana pelinkehityksen maailmassa on koettu ihmeellisiä ja mullistavia hetkiä. Vanhojen pelikehitystyökalujen rinnalle on noussut pikkuruisen tanskalaisstudion luoma Unity 3D, joka nousi ensin indiekehittäjien suosioon ja haastoi sitten perinteiset raskaan sarjan työkalut.

Teksti ja kuvat: Miikka Lehtonen

Vielä muutamia vuosia sitten pelimaailman ykkösvaihtoehto oli Unreal Engine. On tavallaan jopa harhaanjohtavaa puhua ”vaihtoehdosta”, sillä todellisuudessa kaikissa vähänkään isommissa projekteissa Unreal oli ainoa vaihtoehto.

Asetelmat ovat kuitenkin muuttuneet nopeasti ja vieläpä yllättävästi. Unity 3D on kyllä puksutellut jo yhdeksän vuotta. Pienen indieskenen rakastamasta moottorista ja ympäristöstä tuli kuitenkin nopeasti ensin oikea vaihtoehto Unreal Enginelle ja muille suurille moottoreille. Hieman myöhemmin siitä tuli jo markkinaykkönen.

Mistä Unityssä oikeastaan on kyse? Mistä se tuli, mikä tekee siitä niin suositun ja mihin se on menossa – ja minne se vetää perässään koko peliskeneä?

Vallankumous moottorimarkkinoilla

Vielä muutama vuosi sitten pelien kehittäminen oli aika hankalaa puuhaa. Jos ei puhuttu ihan ristinollan tapaisista projekteista, niin monen innokkaan pöytälaatikkokehittäjän ura tyssäsi siihen, että tarvittiin pelimoottori. Se koodirunko, joka huolehti grafiikan piirtämisestä, äänien soittamisesta ja animaatioiden käsittelystä. Se perusta, jonka varaan koko projekti rakentui.

Vaihtoehtoja kyllä oli, mutta niistä useat olivat huonoja. Niin ikään pöytälaatikkokoodarien virittelemät moottorit olivat kankeita ja kömpelöitä, ja pelikehittäjien oli tehtävä vielä valtavasti työtä, ennen kuin varsinainen pelinteko edes pääsi alkuun. Jos tämä ei riittänyt, vaihtoehtona oli joko kehittää itse oma pelimoottori – mikä on hirvittävän hankalaa ja aikaavievää puuhaa – tai maksaa itsensä kipeäksi lisensoimalla ammattilaistyökalut.

Lokakuussa 2009 koettiin pienoinen vallankumous, kun indiemarkkinoilla omaa nurkkaustaan hallinnut Unity Technologies muutti Unity 3D -pelimoottorinsa lisenssiehtoja. Kalliin ammattilaisversion rinnalla oli myyty jo pitkään pienille tiimeille tarkoitettua ja kaikkein kovimmista ominaisuuksista riisuttua indie-versiota, mutta senkin verrattain edulliset lisenssimaksut olivat monelle pöytälaatikkokoodarille ylitsepääsemätön este. Unity Technologies kuitenkin vapautti indie-version vapaaseen levitykseen. Sillä sai tehdä pelejään vapaasti, ja niin myös tehtiin.

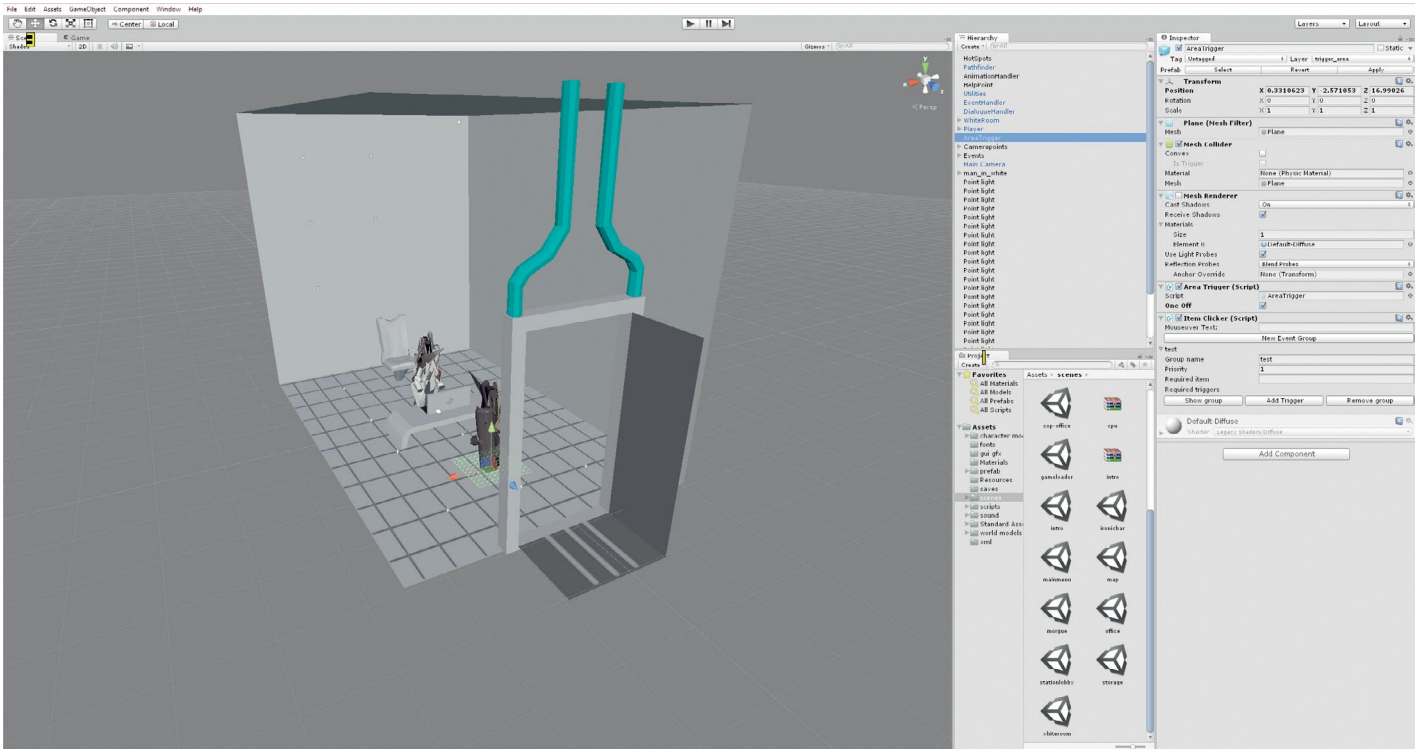
Olin itse juuri Unityn ilmaistumisen aikaan Turun yliopiston pelinkehityskurssilla. Alkuviikkojen aikana miltei kaikki tiimit joko sorvailivat omia moottoreitaan tai yrittivät päästä sinuiksi PyGamen, Python-kielelle rakennetun alkeellisen ja kankean moottorin, kaltaisten viritelmien kanssa. Miltei yhden yön aikana kaikki kurssilla olleet tiimit vaihtoivat Unityn ilmaisversioon, eikä sen jälkeen juuri kukaan ole taakseen katsonut. Lähes kaikki tuolloin Unityyn vaihtaneet ovat myös sen parissa pysyneet.

Emmekä suinkaan olleet ainoat, sillä samaa on tapahtunut kaikkialla muuallakin. Niin pienet muutaman kehittäjän kaveriporukat kuin suuremmat ammattilaisstudiotkin ovat vaihtaneet Unityyn, yksi toisensa perään.

Syyt siihen ovat moninaisia, mutta ennen kuin tutustumme niihin, puhutaan hetki itsestään Unitystä. Mikä tekee siitä niin erikoisen ja suositun?

Unity pähkinänkuoressa

Kehittäjä: Unity Technologies
Kehityksessä: 8.6.2008 lähtien
Tuorein versio: 5.0
Tuetut ohjelmointikielet: Boo, C#, JavaScript
Tuetut kehitysalustat: Mac, Windows
Tuetut kohdealustat: 21 eri alustaa, muun muassa Android, iOS, Linux, Mac, Windows, PlayStation 3, PlayStation 4, PlayStation Vita, Xbox 360, Xbox One ja Wii U



Tyypillinen näkymä. Vasemmalla kohtauseditori. Oikealla selain, jolla voi poimia projektikansiosta assetteja sekä määritellä valitun esineen piirteitä.

Vaikka Unity olikin alunperin vain 3D-käyttöön kehitetty softa – mistä myös nimi – nykyään myös 2D-pelit toimivat sillä erinomaisen hyvin.

Unityn Asset Store tarjoaa mahdollisuuden hankkia grafiikkaa, koodia tai vaikka ääniä peliprojektiinsa, ainakin jos Visasta löytyy vähän vääntöä.

Nopeammin, helpommin, halvemmin

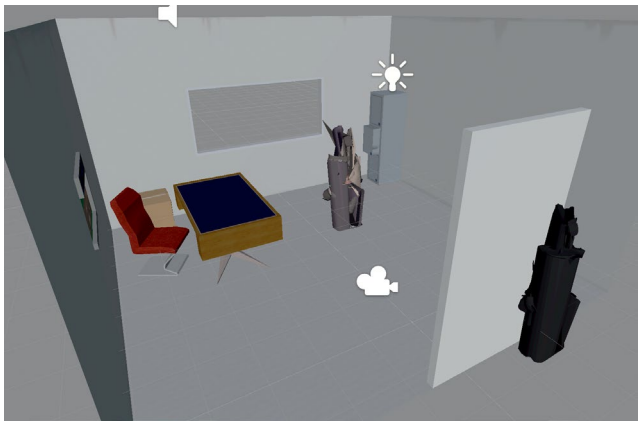
Unity on kattava ja hyvin tuettu pelimoottorin ja kehitysympäristön risteytys. Se soveltuu mainiosti niin 2D- kuin 3D-sovellusten kehittämiseen. Se on tarkoitettu pääasiassa pelimoottoriksi, mutta sillä on toteutettu myös esimerkiksi arkkitehtuurien työkaluja, virtuaalisia kiertoajeluita, interaktiivisia oppaita ja ties mitä muuta. Vuosien varrella Unityyn on lisätty jos jonkinlaisia ominaisuuksia ja se sisältää muun muassa laadukkaat valo- ja fysiikkamoottorit.

Unityn suosion räjähdysmäinen nousu selittyy oikeastaan yhdellä sanalla: helppo. Unity oli monessakin mielessä se ensimmäinen moottori, joka pyrki tekemään asiat helpoksi. Aikanaan historiallisen lempeistä lisenssiehdoista lähtien Unity oli selvästi suunniteltu pelimoottoriksi, joka ei pyrkinyt olemaan elitistinen.

Sen ilmaisversio julkaistiin aikana, jolloin miltei kaikki muut vaihtoehdot sulkivat kehittäjiä ulos joko hinnallaan, saatavuudellaan tai toimivuudellaan. Unity taas pyrki ottamaan kaikki mukaan: asentakaa ilmaiseksi, tehkää pelejä.

Oli toki kova juttu julkaista yleensäkään pelimoottori, jota ihmiset pystyivät käyttämään, mutta sen lisäksi heidät piti saada tarttumaan sorviin ja kokeilemaan. Unity keräsikin huomiota ja mainetta olemalla työkalu, jolla prototyyppailu oli helppoa.

Maine oli ansaittu, sillä Unityllä on



Unityn tiedostotuki on laaja: ääniä, grafiikkaa ja muita assetteja voi hyödyntää melko vapaasti, joskin joskus versioiden väliset yhteensopimattomuusongelmat aiheuttavat päänvaivaa. Näiden kun pitäisi olla ihmishahmoja...

Unityllä tehtyä

Vuosien varrella tyypillinen Unity-peli on tarkoittanut milloin mitäkin. Ominaisuuksien muuttuessa ja suosion kasvaessa "tyypillinen" on kuitenkin muuttunut ehkä harhaanjohtavaksi termiksi, sillä Unityllä on tehty niin suuria kuin pieniäkin pelejä aina muutaman hengen opiskelijaporukoista AAA-tason suuriin julkaisuihin. Tässä muutamia esimerkkejä.

Cities: Skylines – Kyllä, tamperelaisen Colossal Orderin hillittömäksi hitiksi noussut kaupunkisimulaatio pyörii kuin pyöriikin Unityn päällä ja todistaa, että moottori hallitsee hyvin myös suuret asiat.

Kerbal Space Program – Tunnetussa indie-hitissä kasataan palasista avaruusaluksia, joita sitten lauotaan vankan fysiikkamoottorin avulla kiertoradalle – toivottavasti, mutta ei läheskään aina, yhtenä kappaleena!

Grow Home – Myös maailman suurimpiin pelijulkaisijoihin kuuluva Ubisoft käyttää Unityä. Pienen tiimin nopeasta prototyypistä tuli hurmaava peli, jossa fysiikan avulla liikkuva robotti kiipeää jättimäistä pavunvartta pitkin taivaisiin.

Ori and the Blind Forest – Pienen tiimin vuosia vääntämä 2D-tasohyppely on ensimmäisiä Unityllä tehtyjä konsolipelejä. Hämmentävän kaunista grafiikkaa ja suurta pelimaailmaa hyödyntävä peli sai monet Unity-veteraanitkin raapimaan partaansa. "Miten ihmeessä tuo on tehty?"

Hearthstone – Vaikka sitä ei ehkä ensisilmäyksellä uskoisi, Blizzardin mainio ja kymmeniä miljoonia pelaajia viihdyttävä digitaalinen keräilykorttipeli pyörii Unityllä.

Realis3D – 3D-visualisointityökalu on tarkoitettu insinööreille ja arkkitehdeille ja antaa näille mahdollisuuden suunnitella nopeasti erilaisia 3D-ympäristöjä.

aina päässyt nopeasti siihen itse asiaan, eli kokeilemaan ideoitaan ja virittelemään pelejä. Sen keskipisteessä on aikanaan vallankumouksellinen Scene Editor, kohtauseditori, jolla voi helposti ja graafisesti luoda ja käsitellä pelinsä kohtauksia. Oli kyse sitten tasohyppelypelin kentästä tai seikkailupelin huoneesta, oli aivan uskomatonta käyttää pelimoottoria, jossa tilaa pystyi muokkaamaan liikuttelemalla esineitä hiirellä. Kyse on siis hyvin intuitiivisesta ja erittäin miellyttäväkäyttöisestä kehitysympäristöstä.

Nykyään visuaalinen kehitys on toki jo arkipäivää. Täytyy kuitenkin muistaa, että Unity julkaistiin aikana, jolloin piti joko hankalasti toteuttaa moiset ominaisuudet itse tai vaihtoehtoisesti koodata pelejään sijoittelemalla esineitä ja hahmoja maailmaan suorien koordinaattien avulla. Ei kovin kivaa puuhaa tämäkään.

Unityn sujuvuus ei loppunut siihen. Taustalla on toki koodia ja jos siitä haluaa saada täyden hyödyn irti, ohjelmoinnin täytyy sujua. Vallankumouksellista oli se, mitä koodilla sitten tehtiin: graafisessa editorissa pystyi hiirellä raahaamalla liittämään skriptejä ja koodia eri esineisiin. Tämän jälkeen niiden julkisiksi määriteltyjä ominaisuuksia pystyi muokkaamaan suoraan graafisesti ja intuitiivisesti. Tämä oli kova ominaisuus jo niillekin meistä, joilla bitti pysyy hyppysissä, mutta suorastaan vallankumouksellista innokkaille kehittäjille, jotka halusivat suunnitella pelejä, mutta eivät ehkä olleet hirveän taitavia koodareita.

Netissä lähti kiertämään alta aikayksikön ihmisten koodaamia luokkia, joilla liikuteltiin hahmoja, käsiteltiin inventaarioita, ohjattiin autoja ja niin edelleen. Näin innokkaat amatööritkin pystyivät intuitiivisesti ja sujuvasti kokoamaan omia pelejään. Eiväthän ne toki olleet yhtä sujuvia tai hyviä kuin varta vasten koodatut pelit, mutta ne tarjosivat monelle ensimmäisen maistiaisen pelinkehityksestä ja istuttivat halun opetella enemmän.

Ei sovi myöskään unohtaa sitä, että alusta saakka Unityn tarkoituksena on ollut myös tehdä sovellusten kääntäminen eri alustoille mahdollisimman helpoksi. On oikeasti aika kova juttu, että vaikka Windowsilla tehdyn pelin voi pienellä vaivalla ja tuunailulla kääntää Android-puhelimelle tai modernille pelikonsolille.

Kehitys kehittyy, entä moottori?

Unity siis alkoi kerätä itselleen nimeä olemalla tavallaan ensimmäinen. Asiaan toki liittyy myös muita, ulkopuolisia ja ehkä onnekkaitakin sattumia. Merkittävä tekijä sen nousulle on varmasti ollut myös se, että indiepelit ovat yleisestikin viimeisten vuosien aikana keränneet näkyvyyttä. Ne ovat muuttuneet pienen porukan kuriositeeteista yllättävänkin tuottoisiksi myyntihiteiksi.

Sekä mobiili- että tietokonepuolella – ja jopa pelikonsoleilla, kiitos Microsoftin ja Sonyn pelipulan – indie on aina vain kysytympää ja sille riittää markkinapaikkojakin. Niinpä pelialalla oli tilausta pelimoottorille ja työkaluille, joita ei ollut suunniteltu 120 hengen jättitiimien ja kymmenien miljoonien eurojen budjeteilla toteutettavien pelien tarpeisiin, vaan pienille tiimeille ja pieniin peleihin.

Kun putken pää oli avattu, ei sitä tietenkään enää kiinnikään saanut. Unity ei saanutkaan olla kukkulansa kuninkaana kovinkaan pitkään. Epic Games ja monet muut uudemmat ja vanhemmat kehittäjät huomasivat nopeasti, että markkinoilla oli kysyntää skaalautuville pelimoottoreille ja indietiimejä ymmärtäville firmoille.

Niinpä noin vuodesta 2011 lähtien onkin käyty melkoista pelimoottorien kylmää sotaa. Yksi lisää omaan moottoriinsa halutun tuen – vaikkapa työkalut 2D-pelien kehittämistä varten – tai lieventää omia lisenssiehtojaan. Sitten muut seuraavat perässä.

Unity on vuosien varrella kehittynyt ehkä turhankin nopeasti. Vielä tähän kevääseen saakka sen lisenssiehdot, jotka aikanaan olivat niin anteliaat ja kivat, vaikuttivat yllättäen vertailussa suorastaan antiikkisilta. Epicin jaellessa Unreal Engineä ensin halvoilla kuukausimaksuilla ja lopulta "ottakaa nyt vaikka sitten ilmaiseksi!" -hengellä

Unity vaikutti yllättäen kelkasta pudonneelta. Ilmainen harrastelijaversio ei tukenut esimerkiksi lainkaan dynaamisia varjoja, mikä sai kaikki sillä kehitetyt pelit näyttämään suorastaan karseilta. Ikävää oli myös se, että pitkään Unityn parhaat ominaisuudet, kuten pelien porttaaminen eri alustoille, vaativat ylimääräisiä lisenssiostoja ja rahallisia panostuksia. Niihin pienillä tiimeillä ei ollut varaa.

Onneksi lopulta paineeseen vastattiin ja kevään aikana Unitykin pisti lisenssiehtonsa kokonaan uusiksi. Nyt jakelussa on kaksi moottoria, jotka ovat ominaisuuksiltaan identtiset. Ilmaisversiolla voi tykitellä menemään mielensä kyllyydestä, kunhan pelin vuosituotot eivät ylitä sataatuhatta taalaa. Fysiikat, valoefektit ja muut irtoavat ilmaisversiostakin aivan yhtä hyvin kuin ammattiversiosta, joka erottuu paremman ja yksilöidymmän teknisen tuen sekä tiimiominaisuuksiensa avulla.

Nopea kehitys näkyy myös siinä, että Unity kantaa mukanaan yhä paljon koodiriippakiviä. Moottorin parantuessa ja monipuolistuessa monet ominaisuudet ovat olleet yllättävänkin pitkään joko hankalia, kankeita tai rajoittuneita. Moni asia tehtiin aikoinaan paljon simppelimpää käyttöä varten, eikä niitä ole ehditty koskaan tehdä kunnolla uusiksi.

Vaikka Unityn väki onkin tasaiseen vauhtiin tilkinnyt aukkoja ja repinyt riippakiviään modernin ajan puolelle, niin joskus se on kestänyt. Tällöin apu on löytynyt yhteisöstä, sillä Unity on jo pitkään tehnyt tiedon jakelusta helppoa. Omia viritelmiään voi jaella ja tuoda projekteihin helposti. Unityssä on myös ollut jo useamman vuoden ajan virallinen verkkokauppa, jossa voi myydä omia juttujaan. Tarjolla on kaikkea äänistä grafiikkaan ja seikkailupelejä varten tehtyihin skriptausmoottoreihin saakka.

Hillittömän laaja tuki ja suosio tekevät omalta osaltaan Unity-kehittämisestä hyvin mukavaa. Jos jokin asia – kuten vaikka peliohjaintuki – ei Unityssä ole paras mahdollinen, tarjolla on todennäköisesti puolenkymmentä muiden kehittämää plugaria, joilla homma onnistuu juuri niin ammattimaisesti kuin voisi toivoa. Myös Unityn integroituminen koodausympäristöihin on jo itsestäänselvyys. Jos ei halua naputella koodiaan Unityn mukana tulevalla MonoDevelopilla – ja en siitä ketään moittisi – vaihtoehtoista ratkaisua ei tarvitse hakea kaukaa.

Vain taivas rajana?

Unityn esimerkki on innostava. Ei ole montakaan vuotta siitä, kun Unreal Engine oli koko alan standardi. Miltei kaikki varteenotettavat pelit tehtiin sillä, ja Unrealin hallitseminen oli perusedellytys, jos aikoi työllistyä pelialalle. Nyt sama pätee Unityyn.

Jää nähtäväksi, onko edessä uusi vallankumous. Odotteleeko kulisseissa jo jokin uusi pelimoottori, joka pystyy Unityä paremmin vastaamaan vuoden 2015 pelinkehityksen haasteisiin, vai pystyykö moottori mukautumaan pelimaailman tarpeisiin?

En rehellisesti sanoen löisi rahaa Unityä vastaan, sillä se on hämmentävän monipuolinen moottori. Mainiot yhteisöominaisuudet ja helppo laajennettavuus pitävät myös huolen siitä, että vaikka itse moottorin kehittäjät tulisivat huomenna seinähulluiksi, Unityn tie ei tyssäisi siihen.

Vallankumous näyttää siis onnistuneen. Se, mikä oli aikanaan yhtä harvinaista kuin yksisarviset, on nyt alan standardi. Muutkin moottorivalmistajat ovat heränneet kilpailuun. Kun käynnissä on kunnon kilpavarustelu, loppujen lopuksi loppukäyttäjät voittavat, oli oma suosikki Unreal Engine, Unity tai jokin aivan muu.

Tehty Suomessa

Unityä kehitetään nykyään maailmanlaajuisesti, ja myös Suomessa on ollut haaratoimisto jo jokusen vuoden ajan. Suomen pieni Unity-tiimi on todellakin kantanut kortensa kekoon, sillä esimerkiksi Unityn 2D-ominaisuudet ja merkittävä osa nykyisestä käyttöliittymien suunnitteluun tarkoitetusta työkalusta ovat suomalaisten koodaamia.

Raspista vanhan ajan kotitietokone

Tuliko hankittua Raspberry Pi, mutta se on jäänyt alkuinnostuksen jälkeen vähälle käytölle? Oletko kaivannut sitä yksinkertaisuutta ja välittömyyttä, jolla 80-luvun kotitietokoneilla pääsi kokeilemaan BASIC-ohjelmointia? Lähde Skrollin kanssa tutkimaan BBC BASICia!

Teksti: Yrjö Fager
Kuvat: Manu Pärssinen, Yrjö Fager

Raspberry Pi eli Raspi on pieni ja edullinen tietotekniikan ja elektroniikan harrastajille sekä opiskelijoille tarkoitettu tietokone, jonka voi liittää näyttöön HDMI- tai komposiittivideoliitännällä. Raspiin on saatavana ilmaiseksi brittiläisen Acorn Archimedes -tietokoneen käyttöjärjestelmästä kehittyneen RISC OS:n Pico-niminen jakeluversio. Se käynnistää laitteen suoraan BASIC-ohjelmointiympäristöön jäljitellen 80-luvun kotitietokoneiden välitöntä yksinkertaisuutta.

En ollut aiemmin tiennyt BBC BASICista mitään ja törmäsin siihen vasta Raspin kautta. Tässä artikkelissa nähtävä koodi on ensimmäisiä BBC BASIC -ohjelmiani. Toistaiseksi vastaan on tullut lähes pelkästään positiivisia yllätyksiä kielen ja ympäristön suhteen.

RISC OS Picon asennus

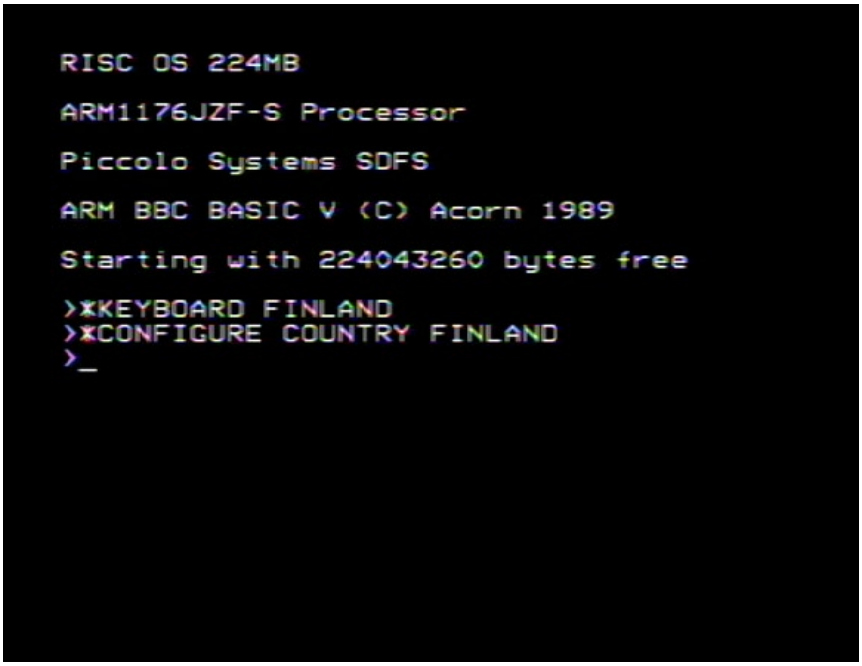
RISC OS Picon voi ladata riscosopen.org-sivustolta. Se asennetaan yksinkertaisesti kopioimalla web-sivuilta ladatusta ZIP-paketista tiedostot FAT-formaattiin alustetulle SD-muistikortille ja laittamalla muistikortti Raspiin.

Kun Raspiin kytkee virran asennuksen jälkeen, laite käynnistyy suoraan BBC BASIC -ympäristöön, ja muuta käyttäjälle ei sitten olekaan tarjolla. Näin konfiguroitu Raspi on selkeä yhden asian laite, koska sillä voi oikeastaan vain harrastaa BASIC-ohjelmointia. Ei hömppäviihdettä tarjoamassa itseään parin klikkauksen päässä, ei nettiselaimia, ei elokuvia, ei edes hauskoja kissavideoita… Tuntuuko vallankumoukselliselta? Enemmän asialle vihkiytynyt henkilö varmaankin kiinnittää muistikortin pysyvästi pikaliimalla.

Käytön aloitus

RISC OS Picon alkuruudussa on varmasti jotain tutun tuntuista 8-bittisiä kotitietokoneita käyttäneille. BASICin käytössä olevan vapaan muistin määrä voi aiheuttaa huvittuneisuutta, se kun on noin 10 000-kertainen niihin lukuihin nähden, joihin 1980-luvulla totuttiin. 200 megatavun muistia ei aivan kevyellä BASIC-koodaamisella saa täyttymään.

Tähtimerkillä alkava komennot `*KEYBOARD FINLAND` ja `*CONFIGURE COUNTRY FINLAND` säätävät laitteen



näppäimistön ja merkistön suomalaisiksi. Jälkimmäinen komento tallentaa asetukset pysyviksi, jotta komentoja ei tarvitse antaa kuin ensimmäisellä käynnistyskerralla. Ensimmäisen komennon tähtimerkki syntyy painamalla Shift+8.

RISC OS Picossa BASIC avautuu oletuksena näyttömoodiin numero 7, toiselta nimeltään Teletext. Moodi on mukavan suuritekstinen, ja omat lapseni ihastuivat isojen kirjainten naputteluun kuvaruudulle. Valitettavasti Teletext-moodissa ei voi kirjoittaa ääkkösiä, vaikka suuremman tarkkuuden ja pienemmän tekstin näyttötiloissa sekä kokoruutueditorissa voi teksti olla aivan oikeaa suomea.

Miten BASIC toimii?

BBC BASIC on käynnistyttyään komentotilassa, jossa komentoja kirjoitetaan >-merkin edustamaan kehotteeseen näppäimistöltä, ja Enter-näppäimellä komennot toteutetaan. Oletuksena Caps Lock on päällä, koska BBC BASIC erottelee isot ja pienet kirjaimet, ja pienellä kirjoitetut sanat eivät tunnistu BASIC-komennoiksi. Esimerkiksi `LIST` toimii käskynä, mutta `list` ja `List` eivät. Pienillä kirjaimilla voi kirjoittaa muuttujien ja aliohjelmien nimiä.

Tässä muutamia komentoja, joita voi aluksi kokeilla:

- `PRINT` tulostaa lainausmerkkien välissä olevan tekstin. Kokeile kirjoittaa `PRINT"MOI"` ja paina Enter.
- `LIST` tulostaa kuvaruudulle nykyisen ohjelman listauksen.
- `RUN` ajaa nykyisen ohjelman. Tietokone suorittaa ohjelmalistauksessa näkyvät rivit alusta alkaen.
- `SAVE` tallentaa ohjelman massamuistiin, Raspin tapauksessa muistikortille, esim. `SAVE"HELLO"`.
- `LOAD` lataa ohjelman massamuistista, esim. `LOAD"HELLO"`.
- `NEW` tyhjentää nykyisen ohjelman.
- `EDIT` avaa ohjelmalistauksen muokattavaksi BBC BASICin sisäänrakennetussa kokoruutueditorissa. Painamalla Shift+F4 pääsee pois editorista ja Ctrl+F5 näyttää ohjeen käytettävissä olevista näppäinyhdistelmistä.

Jos kirjoitat komentorivin alkuun numeron, rivi tallentuu osaksi nykyistä ohjelmaa. Rivinumeroista ei kuitenkaan tarvitse välittää käytännössä ollenkaan, jos käyttää `EDIT`-käskyllä avautuvaa editoria, joka huolehtii rivinumeroista automaattisesti. Samoin on hyvä välttää rivinumeroiden perusteella hyppiviä `GOTO`- tai `GOSUB`-käskyjä ja käyttää niiden sijaan kunnollisia rakenteellisen ohjelmoinnin keinoja.

Jo käytön aloituksessa esitellyt tähtimerkillä alkavat komennot ovat BASICin ulkopuolelle meneviä käyttöjärjestelmätason komentoja. Yksi tärkeimmistä tähtikomennoista on `*CAT`, joka näyttää nykyisen hakemiston tiedostolistauksen.

Ensimmäinen ohjelma

Ensimmäisenä ohjelmana pitää tehdä vanha kunnon Hello World, jotta voi sanoa tehneensä Hello Worldin BASICilla. Kirjoita seuraavat kaksi riviä ja paina tietysti rivien loppuun Enter.

```
10 PRINT"HELLO WORLD"
RUN
```

Onneksi olkoon, olet tehnyt Hello Worldin BASICilla. Heti tämän jälkeen kannattaa siirtyä käyttämään kokoruutueditoria `EDIT`-komennolla.

Eihän BASIC sovi ohjelmoinnin opetteluun?

Toisinaan kuulee väitteitä, että BASIC olisi vanha, huono tai jopa suorastaan haitallinen kieli ohjelmoinnin opetteluun. Väitteiden perusteeksi saatetaan hakea kuuluisien tietotekniikan uranuurtajien 1970-luvulla silloisista ohjelmointitavoista ja -kielistä esittämiä mielipiteitä: `GOTO`-käskyn olemassaolo ja kunnollisen rakenteisen ohjelmoinnin mahdottomuus.

Itse en pidä BBC BASICia ollenkaan huonona tapana tutustua ohjelmoinnin perusteisiin. BBC BASICissa rivinumeroista ei tarvitse välittää, `GOTO`-käskyä ei tarvitse käyttää ja käytössä ovat rakenteellisen ohjelmoinnin peruskeinot: nimetyt aliohjelmat parametreineen ja paikallisine muuttujineen, aliohjelmien rekursio, `FOR`-silmukat, `REPEAT`- ja `WHILE`-toistolauseet sekä rakenteiset tietotyypit. Kasvunvaraa löytyy tarvittaessa vaikka ARM-assemblerin tai omatekoisten oheislaitteiden GPIO-ohjauksen suuntaan.

BASIC-ohjelmointia, saati sitten ohjelmointia yleisesti, ei voi tällaisessa artikkelissa opettaa, mutta yritän antaa siitä pienen maistiaisen.

Demoefekti BBC BASICilla: twisteri

Yleensä ohjelmoinnin alkeiskurssien opiskelijoiden oletetaan innostuvan, jos esimerkkiohjelmana on jonkinlainen peli. Itse en ole aivan hirveän innoissani pelien ohjelmoinnista enkä pelaamisesta, vaan otan ohjelmaesimerkiksi twisteriksi kutsutun perinteisen demoefektin, joka on mielestäni varsin hupaisan näköinen otus ja suoraviivainen ohjelmoida.

Twisteri on akselinsa ympäri pyörivä elastiselta näyttävä kappale, usein yksinkertainen halkaisijaltaan neliön

muotoinen palkki. Sitä piirretään kuvaruudulle suikaleittain siten, että kukin suikale on eri pyörähdyskulmassa. Pyörähdyskulman muutoksen suuruus suikaleiden välillä ja animoinnin kuvien eli ”freimien” välillä määrää sen, miten nopeasti ja mihin suuntaan palkki pyörii ja miten tiukalle kierteelle se näyttää vääntyvän. Nopeuksia vaihtelemalla saa aikaan erilaisia väänteleyitä ja pienillä lisäyksillä kaikenlaisia muitakin efektejä.

Edistyneessä twisterissä voi olla vaikkapa pintatekstuuria, se voi heittää itseensä varjoja ja palkin tahkot voivat olla erilaisia. Eikä kappaleen tarvitse olla suorakulmainen, vaan se voisi olla jotain paljon monimutkaisempaa, jolloin aletaan lähestyä jelly-vektoreiksikin kutsuttuja efektejä.

Kulmapisteiden x-koordinaattien laskenta

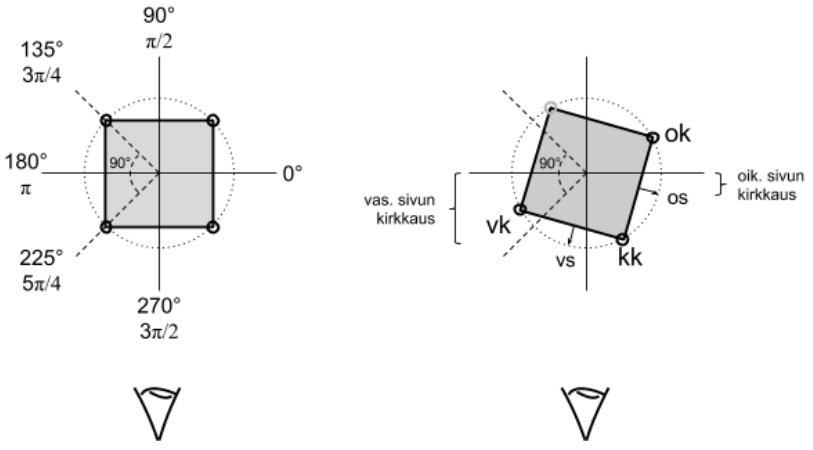
Meidän BASIC-twisterimme on mallia ”basic”: halkaisijaltaan neliön muotoinen harmaa palkki, jota piirretään kuvaruudulle yhden pikselirivin kokoisina suikaleina.

Perusteellisesti tehdyssä neliötwisterissä kierrettäisiin neliön kaikkia neljää kulmapistettä 2D-rotaatiomatriisilla ja jokaisesta sivusta tutkittaisiin erikseen, onko se kääntyneenä katsojaan päin ja siis näkyvissä vai katsojasta poispäin ja piilossa. Teen asian hiukan toisin: koska jokaisessa pyörähdysasennossa on joka tapauksessa enintään kaksi sivua näkyvissä ja kaikki sivut ovat keskenään samanlaisia, ei meitä kiinnosta, mitkä kaksi ne ovat. Kappaleen symmetrian ansiosta riittää, että osataan käsitellä yksi 90 asteen eli puolikkaan piin levyinen väli kärkipisteiden pyörimiskulmista.

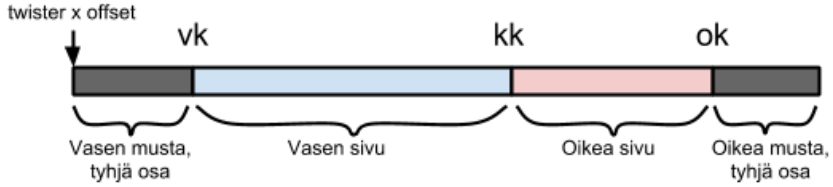
Kun neliön pyörähdyskulmasta otetaan modulo- eli MOD-operaattorilla jakojäännös 90 asteella jakamisesta, saadaan sen kulmapisteen pyörähdyskulma, joka näkyy ruudulla vasemmanpuolimmaisena. Keskimmäinen kulmapiste on tästä pisteestä 90 asteen päässä ja oikeanpuoleinen tästä saman pyörähdysetäisyyden päässä. Kosinifunktio kustakin näistä kulmista kertoo kyseisen kulmapisteen vaakasuuntaisen eli x-koordinaatin suhteessa neliön keskipisteeseen.

Kuva 1 yrittää havainnollistaa ylhäältä päin katsottuna neliön kärkipisteiden ja sivujen normaalien pyörimistä sekä vasemmanpuolisen näkyvän kärkipisteen löytymistä kulmaväliltä $\frac{3}{4}\pi - \frac{5}{4}\pi$. Kuva 2 esittää edestäpäin katsottuna kärkipisteiden vk, kk ja ok kuvautumisen kuvaruudulle piirrettävään suikaleeseen.

Joku saattaa ihmetellä sitä, että laskemme suikaleista pelkät x-koordinaatit. Periaatteessa voisi ajatella, että kärkipisteiden etäisyyden katsojasta tulisi jotenkin näkyä, mutta käytännössä twisteri näyttää aivan hyvältä ilman etäisyyden huomioimistakin. 3D-grafiikkaa tai valokuvausta harrastanut lukija osannee antaa ilmiölle uskottavalta kuulostavan teoreettisen perustelun, jossa saatetaan mainita sanat perspektiivi ja polttoväli.



Kuva 1: Pyörimiskulmat, kärkipisteet ja sivut.



Kuva 2: Suikaleen osat.

Pinnan valaistus

Pinnan valaistuksen laskemme yksinkertaisella periaatteella: otetaan sivun normaalia edustamaan kulmapisteiden puolivälistä kulma ja siitä sinifunktio, joka muuntaa sivun normaalia edustavan kulman kirkkausarvoksi. Kun tähän lisätään vielä pieni kulmasta riippumaton vakio, saadaan simuloitua tilannetta, jossa kappaleen materiaali on mattapintainen ja valon heijastuminen täysin ”diffuusi”, valo tulee katsojan suunnasta ja lisäksi ympäristössä on kulmasta riippumattoman vakion suuruinen ambienttivalaistus.

Yleisesti diffuusin heijastuksen voimakkuus lasketaan valaistun pinnan normaalivektorin ja valovektorin välisenä pistetulona. Koska tässä tapauksessa valo on sopivasti katsojasta kappaleeseen päin olevan akselin suuntainen ja pinnan normaali on valmiiksi sini- ja kosinifunktioihin syötteeksi kelpaavan kulman muodossa, laskukaava yksinkertaistuu.

Listauksessa 1 ambienttivalon vakiona on sivujen kirkkauden laskentakaavassa numero 35, ja heijastuksesta tulevan kirkkauden osuudelle annetaan sinifunktion kertoimeksi 220 yksikköä. Näistä yhteen laskettuna tulee korkeintaan 255, joka on suurin mahdollinen 8-bittiseen värikomponenttiin mahtuva arvo. Jos summa voisi olla suurempi, kirkkausarvo täytyisi jotenkin rajoittaa, jolloin osa kirkkauksista leikkautuisi maksimiin.

Ensimmäisen ohjelmaversion toiminta

Listauksessa 1 on puhtaasti BBC BASIC -käskyillä tehty twisterimme ensimmäinen versio, joka laskee tunnollisesti kaikki matemaattiset laskutoimitukset jokaisen kuvan jokaiselle suikaleelle trigonometrisine funktioineen päivineen ja piirtää kuvan ruudulle PLOT-käskyillä. Kuvassa 3 on neljä kuvakaappausta twistauksen eri vaiheista.

PLOT-käskyjen syntaksi on hieman vaikeaselkoinen, koska ensimmäisenä parametrina oleva numero määrää käskyn varsinaisen tyypin. PLOT 100 siirtää nykyisen piirtokohdan annettuun kohtaan ruudulla. PLOT 97 piirtää parametrina annetun kokoisen täytetyn suorakaiteen suhteellisesti piirtokohtaan nähden ja siirtää piirtokohtaa annetun koon verran. GCOL-käsky vaihtaa piirtoväriksi annetun RGB-arvon. Piirtokäskyjen numeroiden ja parametrien luettelot löytyvät BBC BASIC -ohjelmointioppaista.

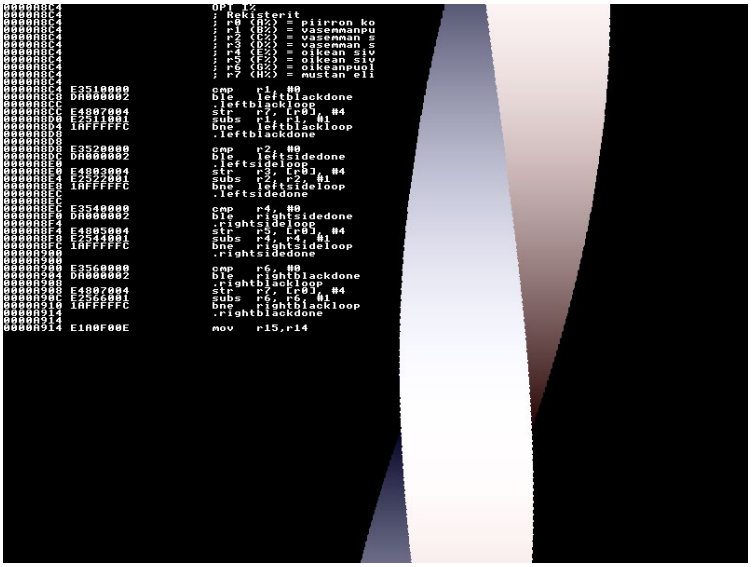
Toisin kuin 2D-tietokonegrafiikassa yleensä, BBC BASICin grafiikkakoordinaatistossa origo eli piste (0,0) on kuvaruudun vasemmassa alakulmassa ja Y-koordinaatti kasvaa ylöspäin niin kuin koulutunneilta tutussa matemaattisessa koordinaatistossa. X-koordinaatti kasvaa normaalisti oikealle. Lisäksi koordinaattiyksiköt eivät ole suoraan pikseleitä vaan ”loogisia yksikköjä”. Jokaista kuvaruudun pikseliä vastaa kaksi loogista yksikköä, eli loogisten yksikköjen piste (100,100) onkin kuvaruudun pikseli (50,50). Siksi ohjelmassa koordinaatteja kerrotaan 2:lla.

Rutiinin nopeuttamista BASICin keinoin

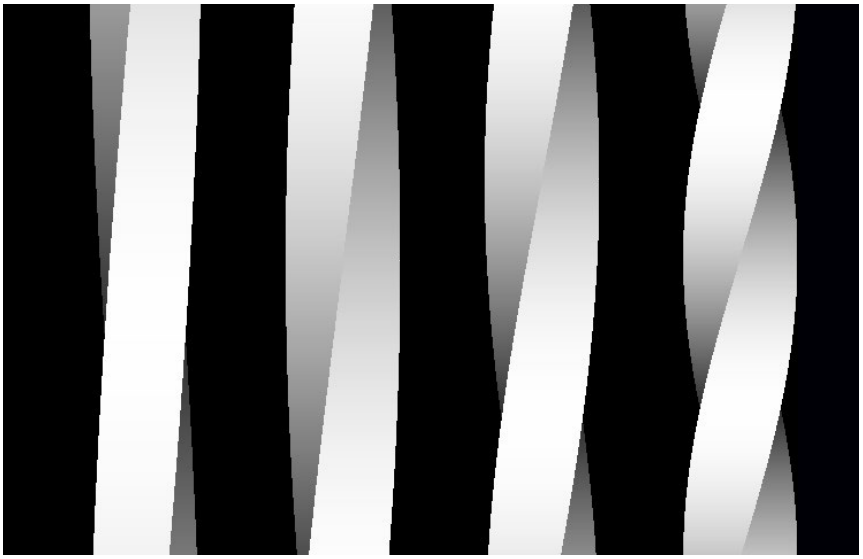
BASIC-versio piirtyy käyttämälläni Raspberry Pi model B:llä nopeudella 20 kuvaa sekunnissa. Se on ihan mukavalta tuntuva nopeus BASICilla tehdyksi efektiksi, ainakin jos vertaa siihen, minkälaisia graafisia efektejä 80-luvulla BASICilla tehtiin. Raspin satojen megahertsien taajuudella toimivan 32-bittisen prosessorin pitäisi kuitenkin kaiken järjen mukaan pystyä paljon parempaankin. On siis aika tutkia, saisikohan efektiä jotenkin nopeutettua.

Kokeilin laskea etukäteen eli ”precalcata” kaikki piirron PLOT-rutiinien tarvitsemat luvut valmiiksi taulukkoon. Sitten poistin koko PROCtwister_slice-aliohjelman siirtämällä piirtorutiinien kutsut suoraan pääohjelman FOR-silmukkaan. Tällä sain nopeuden nostettua 30 kuvaan sekunnissa, mutta sekään ei vielä ollut tyydyttävä tulos. Demoefektien kuuluu liikkua sulavasti täydellä kuvaruudun päivitystaajuudella. Kun kommentoin PLOT- ja GCOL-piirtokäskyt piiloon, nopeus nousi lähes 200 kuvaan sekunnissa. Siksi päättelin, että piirtokäskyt olivat ohjelmassa pullonkaula. Nämä versiot jätin lehdestä pois.

Maksimipiirtonopeuden mittaamista varten otin pääsilmukasta pois WAIT-käskyllä tapahtuvan kuvaruudun päivitystahtiin synkronoinnin eli ”vsyncin”, jotta nopeus ei pyöristy lähimpään ruudunpäivityksen keston monikertaan. WAIT-käskyn kanssa mitään 200 kuvan sekuntinopeutta ei pystyisi saamaan, jos ruudunpäivitystaajuus on vaikkapa 60 kuvaa sekunnissa.



Kuva 4: Kuva twisterin konekieliversiosta.



Kuva 3: Kuvasarja twisterin BASIC-versiosta.

Tule apuun, assembler

Koska piirtokäskyt näyttivät olevan pahin nopeutta rajoittava tekijä, ja koska arvioni mukaan Raspin prosessorin pitäisi ehtiä piirtää suikaleet aivan kevyesti haluamassani ajassa, otin järeämmät keinot käyttöön. Jätin PLOT-kutsut kokonaan pois ja kirjoitin suikaleen piirtämisen pieneksi assembly-rutiiniksi. Tämä oli yllättävän helppoa, kun yhdellä nettifoorumilta löytämälläni systeemikutsulla sain käyttöön kuvaruudun muistipuskurin, ja assembly-rutiinikin meni oikein jo toisella kokeilulla.

Ensimmäisellä ajoyrityksellä ruudulle ilmestyi vain yksi suikale ja jokin hiukan oudompi virheilmoitus. Tämän jälkeen BASIC-ympäristö ei enää käyttäytynyt normaalisti ja tarvitsi uudelleenkäynnistyksen. Onneksi olin varautunut tähän tallentamalla ohjelman uudella nimellä jokaisen muutoksen jälkeen.

Kaatumisen syy oli yksinkertainen: olin unohtanut kirjoittaa rutiinin loppuun käskyn mov r15,r14. Käsky sijoittaa paluuosoitteen sisältävän link-rekisterin (r14) arvon prosessorin nykyistä ajettavaa käskyä osoittavaan ohjelmalaskuriin (Program Counter, r15) eli hyppää takaisin sinne, mistä aliohjelmaa kutsuttiin. Paluukäskyn puuttuessa prosessori jatkoi viimeiseksi tarkoitetusta käskystä ohi jonnekin,

mitä siellä sitten sattuikaan muistissa olemaan.

Optimoitu efekti pyöri jo niin vaivattomasti ”frameen”, että kasvatin twisterin kokoa ja näyttötilan tarkkuutta ja lisäsin pieniä lisähienouksia. Assembly-versiossa twisterin sivuilla näyttävät loistavan värikkäät valot: vasemmalla sininen ja oikealla punainen valo. Tämä versio on listauksessa 2, joka löytyy osoitteesta skrolli.fi/2015.2. Listauksessa on assembly-rutiinin lisäksi huomionarvoinen `DIM`-käskyllä varattava taulukko `twister_slices%`, johon ohjelma laskee valmiiksi kaikkien eri pyörähdyskulmissa olevien suikaleiden piirtämiseen tarvittavat parametrit.

Kuvassa 4 on kuvakaappaus assembly-versiosta. Vasta pysäytetystä kaappauksesta huomasin, että twisterin reunoissa näkyy pientä sahalaitaisuutta, joka ehkä johtuu liukulukujen pyöristysvirheistä. Twisterin vieressä näkyy assemblerin tuottama tekstituloste.

Jotain lisättävää?

Assembly-rutiinin piirtämän taustavärin RGB-väriarvo menee rutiinille parametrina, joten siinä ei tarvitsisi antaa välttämättä nollaa eli mustaa. Jos suikaleiden reunoilla olevat taustaväriset eli mustat osat levittää koko ruudun levyisiksi, saisi taustaväriä jokaiselle vaakajuovalle erikseen säätämällä twisterin taakse vaikkapa perinteiset liikkuvat väripalkit tai ihan yksinkertaisen liukuvärin. Entä teksturointi: mitä ohjelmaan pitäisi tehdä, jos haluaisi twisterin pinnan näyttävän vähemmän sileältä? Väriarvoilla ”fillaamisen” sijasta suikaleen piirtorutiini voisikin kopioida tekstuuria jostakin valmiiksi lasketusta bittikarttataulukosta. Tai voisiko pinnan heijastus olla vähemmän diffuusi, tai voisivatko valonlähteiden värit muuttua lennossa? Nämä muutokset jätämme lukijoiden askarreltaviksi.

Olemmeko nyt eliittikoodareita?

Emme ehkä tulleet tehneeksi aivan näyttävintä mahdollista Raspi-efektiä, koska ohjelmamme ei ollenkaan hyödynnä esimerkiksi Raspin GPU-piiriä. Siltä twisterin piirtäminen kävisi huomattavasti eleettömämmin. Vai mitä pitäisi ajatella siitä, että 25 vuotta vanhemmalla Amiga 500:lla pystyy tekemään käytännössä täysin samanlaisen efektin?

Amigan noin 7 MHz:n Motorola 68000 -prosessorissa on laskentatehoa hädin tuskin yhtä tuhannesosaa Raspin ARM-prosessoriin verrattuna, mutta pääprosessorin ei tarvitsekaan tehdä juuri mitään. Twisteri käy Amigalta vaivattomasti, jos ohjelma piirtää eri kulmissa olevat suikaleet valmiiksi bitmap-kuvaan ja vaihtelee kuvaruudulle piirtyvän alueen osoitetta jokaiselle vaakajuovalle erikseen Copper-piirin avulla. Amigan Copper- ja Blitter-apupiirejä voi pitää jossain mielessä GPU-tyyppisenä ratkaisuna. Apupiirit suorittavat asioita itsenäisesti pääprosessorista riippumatta ja toimintaa pystyy ohjaamaan kirjoittamalla copperlistoiksi kutsuttuja ohjelmanpätkiä yksinkertaisella komentokielellä.

```
REM *** Näyttötilan asettaminen ***
screen_width% = 640
screen_height% = 480
MODE screen_width%, screen_height%, 32

REM *** Muuttujien alkuasetukset ***
twister_height% = screen_height%
twister_width% = 200
twister_radius% = twister_width% DIV 2
twister_xoffset% = screen_width% - twister_width%
rotation% = 0
twistratiophase = 0
framecount% = 0
starttime% = TIME

REM *** Pääsilmukka toistuu, kunnes käyttäjä painaa näppäintä ***
WHILE INKEY(0) = -1
  twist = 0
  twistratio = SIN(twistratiophase) * 2

  REM Odotetaan kuvaruudun päivitystä eli "vsync"
  WAIT

  FOR Y% = 0 TO twister_height% - 1
    PROCtwister_slice(Y%*2, twister_xoffset%*2, rotation%+twist)
    twist = twist + twistratio
  NEXT Y%

  rotation% = rotation% + 40
  twistratiophase = twistratiophase + 0.03
  framecount% = framecount% + 1
ENDWHILE

endtime% = TIME
PRINT (framecount%*100)/(endtime%-starttime%) " frames per second"

END


REM *** Aliohjelma, joka piirtää yhden suikaleen ***
DEF PROCtwister_slice(row%, xoffset%, angle%)
  LOCAL leftx%, midx%, rightx%
  LOCAL light1%, light2%
  LOCAL modangle

  REM Välillä 3/4 PI .. 5/4 PI oleva piste on
  REM aina vasemmanpuoleisin
  modangle = (angle% MOD 512 + 768) * PI / 1024.0
  leftx% = COS(modangle) * twister_radius%
  midx% = COS(modangle + PI / 2) * twister_radius%
  rightx% = COS(modangle + PI) * twister_radius%

  REM Näkyvissä olevien kahden sivun kirkkaus
  light1% = -SIN(modangle + PI / 4) * 220 + 35
  light2% = -SIN(modangle + 3 * PI / 4) * 220 + 35

  REM Piirretään sivut näkyviin
  GCOL 0, 0, 0
  PLOT 100, xoffset%, row%
  PLOT 97, twister_radius%+leftx%, 0
  GCOL light1%, light1%, light1%
  PLOT 97, midx%-leftx%, 0
  GCOL light2%, light2%, light2%
  PLOT 97, rightx%-midx%, 0
  GCOL 0, 0, 0
  PLOT 97, twister_radius%-rightx%, 0
ENDPROC
```

Listaus 1: 100 % BASIC-versio

Ehkäpä twisteriefektimme kuitenkin havainnollistaa jotakin Raspin ja BBC BASICin mahdollisuuksista ja kukaties innostaa jotakuta kokeilemaan ohjelmointia.

Lisätietoja aiheesta

RISC OS Picon mukana tulee PDF-versiona Martyn Foxin kirjoittama kirja *First Steps in Programming RISC OS Computers*, joka kertoo BBC BASIC -ohjelmoinnista erittäin laajasti aina ARM-assembleriin saakka. Sen lisäksi riscosopen.org-sivustolta löytyy runsaasti tietoa ja aktiivisen tuntuiset käyttäjäfoorumit, joilla näyttää olevan paljon Raspberry Pi -käyttäjiä.

Katso listaus 2: http://skrolli.fi/2015.2

Varmuuden hinta

Samalla kun kokemusmaailmamme sähköistyy, tietoturvan merkitys korostuu koko ajan. Onneksi käyttäjien, laitteiden ja tiedon suojaamiseksi on olemassa joukko erilaisia tekniikoita. Yksi tärkeä työväline suojaukseen ovat erilaiset varmenteet eli sertifikaatit.

Teksti: Outi Sorsa

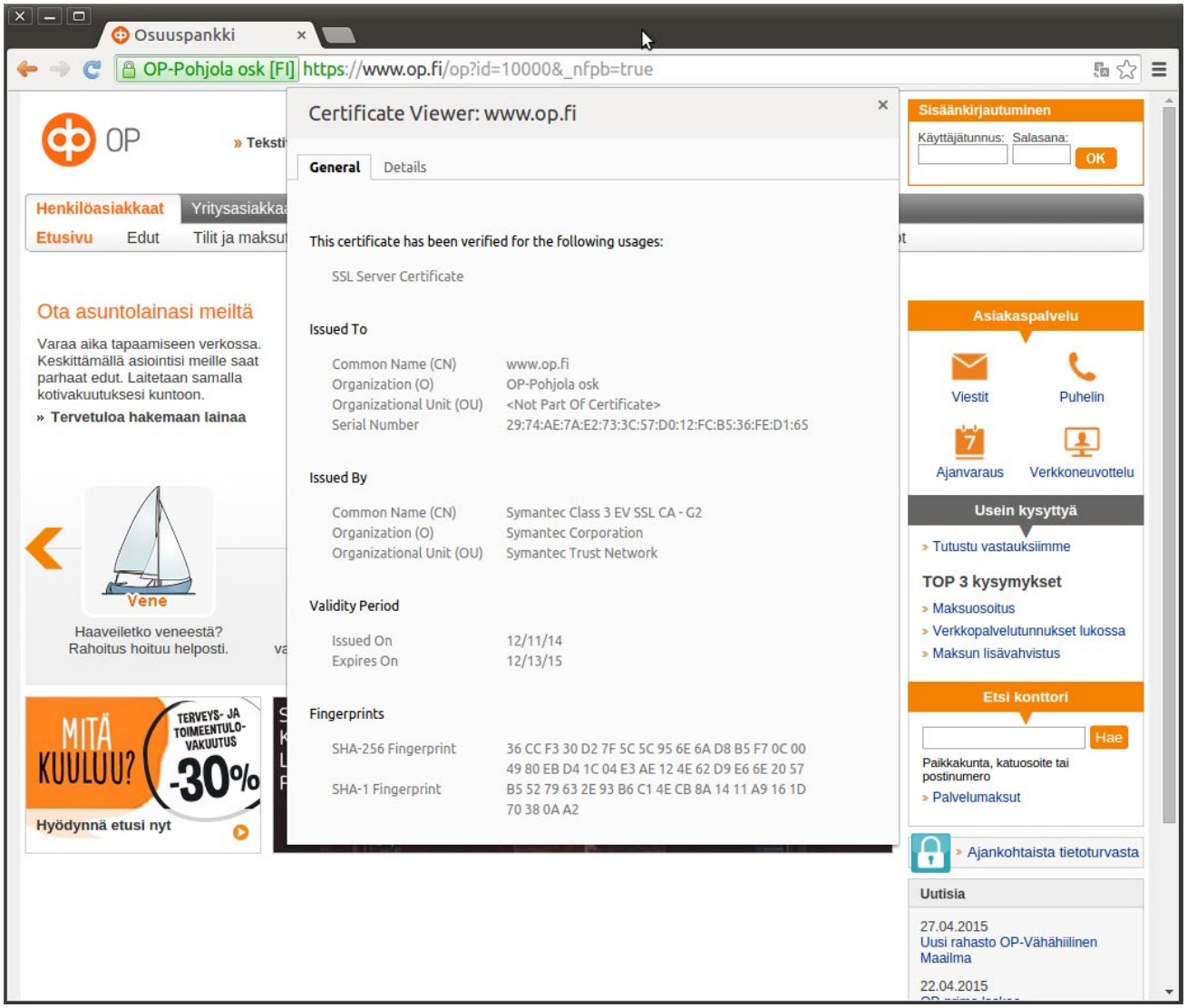
Kuvat: Mitol Meerna, Outi Sorsa

Internet on osa nykyajan arkipäivää. Sen avulla ollaan yhteydessä ystäviin ja perheenjäseniin, asioidaan pankissa, tehdään ostoksia ja ohjataan talojen valaistusta ja lukkoja. Suurin osa käyttäjistä ei kuitenkaan ole kunnollisesti perillä verkon tietoturvariskeistä. Onneksi käyttäjien, laitteiden ja tiedon suojaamiseksi on olemassa joukko erilaisia tekniikoita. Yksi tärkeä työväline suojaukseen ovat sertifikaatit.

Ennen internetiä ihmiset kävivät nostamassa rahansa pankista ja kauppoihin käveltiin katsomaan tuotteita ja ostamaan. Mainokset oli painettu paperille ja niissä kerrottiin kauppojen fyysiset osoitteet. Pystyimme siis käsin koskettamaan rahoja maksutilanteessa, toteamaan missä kaupassa olimme ja kuka meitä palveli. Enää tilanne ei ole aina sama. Suurin osa maksuliikenteestä kulkee sähköisesti ja paljon ostoksia tehdään sähköisesti, internetin välityksellä. Tämä vaatii hyvin paljon luottamusta kuluttajalta. On luotettava siihen, että tuote on se joka halutaan, rahat siirtyvät oikein ja kauppa on oikeasti olemassa. Tämän lisäksi meidän pitää pystyä varmistumaan siitä, että kauppatilanteessa yhteys on juuri oikeaan kauppaan, eikä kukaan ole kaapannut kuluttajan ja kaupan välistä liikennettä.

Salausmenetelmät

Tärkeä osa tietoliikennettä on liikenteen salaus. Emme halua, että kukaan ulkopuolinen voi lukea lähettämiämme viestejä, joten haluamme piilottaa viestin tai muuttaa sen niin, etteivät



Esimerkki sertifikaatista.

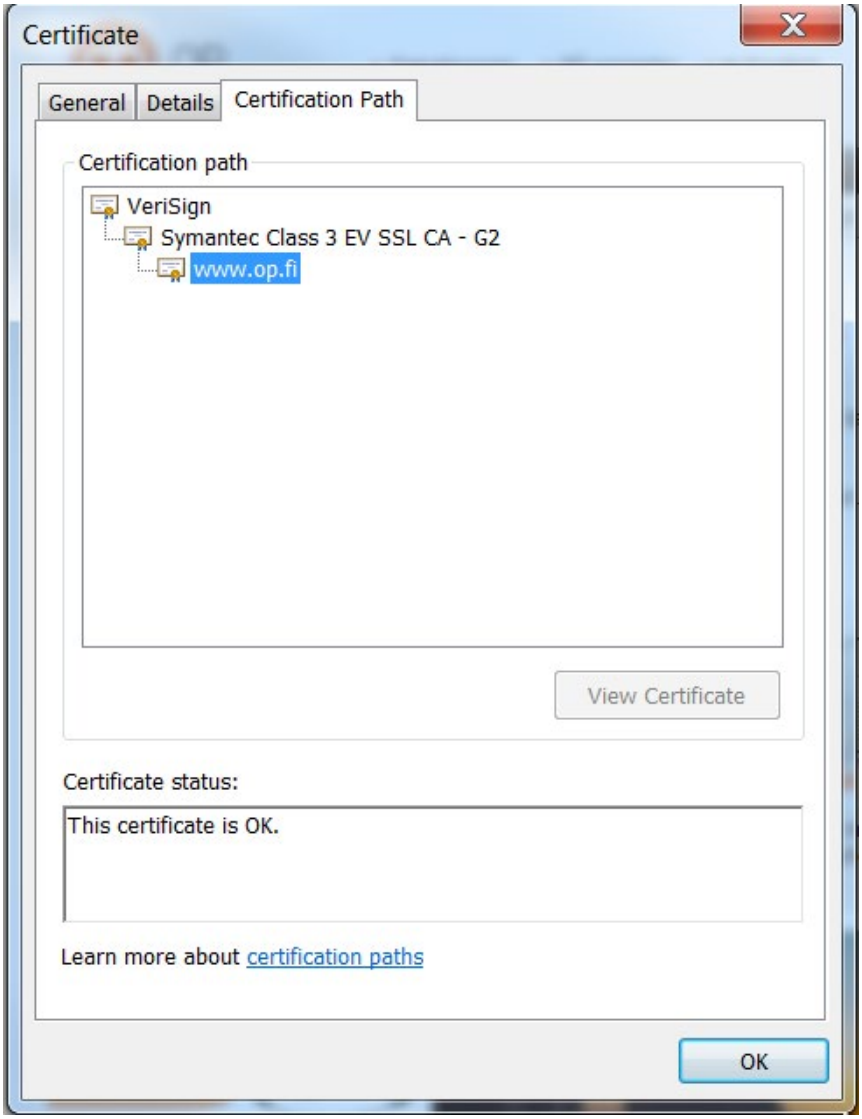
ulkopuoliset voi sitä ymmärtää. Ensimmäinen tunnettu salausmenetelmä on Caesarin käyttämä salakirjoitustekniikka, jossa kirjain korvataan 3 kirjainta aakkosjärjestyksessä eteenpäin olevalla kirjaimella. Nykyisin käytössä ovat kehittyneemmät menetelmät. Niistä tunnetuimpiin kuuluvat symmetrinen salausalgoritmi AES ja epäsymmetrinen RSA.

Symmetrinen salausalgoritmi tarkoittaa sitä, että sekä lähettäjä että vastaanottaja tarvitsevat saman avaimen viestin lukemiseen. Samalla avaimella voi siis sekä salata että purkaa salauksen. Tämä tekniikka toimii hyvin, jos sama avain saadaan jaettua lähettäjälle ja vastaanottajalle ennen yhteyden muodostusta. Esimerkiksi langattoman lähiverkon tukiasemaa asennettaessa voidaan yhteyden salaustavaksi valita AES eli Advanced Encryption Standard. Tällöin verkon salasanan voi jakaa vaikka suullisesti käyttäjille.

Epäsymmetrinen salaus taas tarkoittaa sitä, että tarvitaan kaksi erillistä avainta, toinen viestin kirjoitukseen ja toinen sen purkuun. Yleensä toinen näistä avaimista on julkinen ja toinen salainen. Avaimet on valittu niin, että niillä on matemaattinen riippuvuus toisiinsa, kuitenkin niin että lukuparin toinen luku on lähes mahdoton ratkaista pelkästään toisen luvun avulla tai ratkaisu kestää hyvin pitkään. Tätä kutsutaan yksisuuntaiseksi algoritmiksi.

Jos epäsymmetrisen salauksen salausavain julkaistaan, voivat muut lähettää salattuja viestejä avaimen omistajalle ja vain avaimen omistaja voi purkaa viestit ja lukea ne. Jos taas purkuavain julkaistaan, voi kuka tahansa purkaa avaimen omistajan viestejä, mutta samalla varmistua viestin lähettäjästä. Tässä tapauksessa vain oikea avain purkaa viestin. Voimme siis varmistua siitä, että avaimen alkuperäinen omistaja on viestin lähettäjä. Tätä tapaa käytetään hyväksi digitaalisessa allekirjoituksessa.

Digitaalista allekirjoitusta käytetään hyväksi internetissä paljonkin. Haluamme varmistua siitä, että viesti on aito ja sitä ei ole käsitelty sen jälkeen, kun se on luotu. Yleensä viestin julkaisija tekee viestistä tiivisteen käyttämällä yksisuuntaista salausalgoritmia, kuten SHA-256. SHA (*Secure Hash Algorithm)* on yksi tunnetuimmista tiivistefunktioista, jota käytetään useissa protokollissa ja ohjelmistoissa. Tiivisteet nimensä mukaisesti tiivistävät viestin lyhyemmäksi merkkijonoksi. Joissain tapauksissa merkkijono voi myös kasvaa, jos se on todella lyhyt. Tiivistystavasta riippuen merkkijono on usein saman mittainen. Eli esimerkiksi tuhannen sanan essee ja viiden



Kuvassa varmenteen antajana (Certificate Authority) VeriSign.

kirjaimen sana voivat molemmat tuottaa 15 merkin jonon. Näin ollen olemme luoneet lyhyen merkkijonon, joka muuttuu aina, jos viestiä muutetaan.

Alkuperäisen viestin mukana julkaistaan tiivistesumma, ja vastaanottaja voi testata tiivisteen paikkansapitävyyttä kokeilemalla tiivistää viestin uudelleen. Jo yhden kirjaimen muutos muuttaa tiivistesumman täysin, joten voimme huomata, jos joku on muuttanut viestiä. Kun vielä salaamme tämän viestin epäsymmetrisellä salausmenetelmällä ja julkaisemme purkuavaimen, voi vastaanottaja varmistua, että viestin sisältö on pysynyt koskemattomana ja lähettäjä on se, joka on salannut viestin. Tätä kutsutaan digitaaliseksi allekirjoitukseksi.

Digitaalinen allekirjoitus ei kuitenkaan yksinään riitä. Mistä voimme olla täysin varmoja, että joku ei ole purkanut välissä viestiä ja salannut sitä uudelleen ja antanut meille uudelleen salatun tiedon julkisen avaimen? Jotta voimme varmistua toisen osapuolen identiteetistä, käytämme hyväksi kolmatta tahoa, jota pidämme luotettavana. Avaimen omistaja voi ostaa sertifikaatin, eli tavallaan digitaalisen henkilöllisyystodistuksen. Kolmas taho luo siis digitaalisen dokumentin, jonka se itse allekirjoittaa. Digitaalinen dokumentti pitää sisällään tiedot avaimesta ja sen omistajasta (muun muassa fyysisen osoitteen), ja sertifikaatin myöntäjä vielä allekirjoittaa sertifikaatin digitaalisesti.

Viestin lähettäjä siis ensin tiivistää viestinsä ja liittää tiivisteen alkuperäiseen dataan, jotta voimme varmistua, ettei tietoa ole käsitelty. Viesti salataan epäsymmetrisellä salauksella, jotta vastaanottaja tietää, että viesti on juuri sen salanneelta taholta. Tämän lisäksi viestiin liitetään vielä sertifikaatti, joka varmistaa että viestin salaaja on se, joka sanoo olevansa.

Ketjutettuja sertifikaatteja ja kaappaajia

Suurin ongelma on, että kuka tahansa voi luoda sertifikaatteja. Eli käyttäjän tulisi olla kiinnostunut siitä, kuka sertifikaatin on myöntänyt. Monet selaimet pitävät nykyisin sisällään tiedon siitä, mitkä yritykset ovat luotettavia sertifikaattien luojia, niin kutsuttuja Certificate Authority (CA). Tässäkin on kaksi tasoa. On niin sanotut juuri-CA:t, eli suurimman luottamustason yritykset, ja välitason CA:t. Välitason CA:t ovat yrityksiä, joille juuri-CA on allekirjoittanut sertifikaatit. Sertifikaatit voivat siis myös ketjuttua useammille osapuolille. Jos selain tai muu päätelaite ei löydä sertifikaattia omista tiedoistaan, se alkaa tutkia, onko sertifikaatin myöntäjä joku luotettu taho. Jos sertifikaatin myöntäjää ei löydy listasta tai se on vanhentunut, näemme varoituksen että sivu ei ole ehkä luotettava. Suurin osa ihmisistä jatkaa kuitenkin sivulle.

Sertifikaatit ovat oiva apu torjua niin kutsuttuja Man-in-the-Middle-hyökkäyksiä. Suomalaisittain termi on välistäveto- tai mies välissä -hyökkäys. Man-in-the-Middle-hyökkäyksessä hyökkääjä varastaa kahden osapuolen välisen liikenteen ja teeskentelee molemmille olevansa vastakkainen osapuoli. Näin hyökkääjä voi saada tietoonsa esimerkiksi luottokorttien numeroita, käyttäjätunnuksia ja salasanoja. Mahdollisesti hyökkääjä voi myös muuttaa matkalla viestin tietoja, kuten pankkimaksun summaa ja tilinumeroa.

Hyvin harva peruskäyttäjä oikeasti tietää, mitä tulee tehneeksi, kun ohittaa selaimen varoitukset sivusta, jonka varmenteet ovat vanhentuneet tai niissä on jotain väärin. Suurimmalle osalle varoitus on vain ärsyttävä sivu, joka tulee ennen kuin pääsee esimerkiksi verkkokauppaostoksille. Varoitukselle voi kuitenkin olla syynsä.

Oculus Rift -ohjelmointi

Virtuaalitodellisuuteen sopivat silmikkonäytöt ovat kovaa kyytiä tulossa markkinoille. Mutta kuinka niitä virtuaalimaailmoja sitten luodaan? Se ei onneksi ole vaikeaa.

Teksti: Mikko Rasa
Kuvat: Sakari Leppä, Mikko Rasa

Tässä artikkelissa esittelemme matalan tason lähestymistavan virtuaalitodellisuuden ohjelmointiin käyttäen OculusVR:n julkaisemaa LibOVR-kirjastoa. Pohjana käytämme OpenGL-ohjelmointiartikkelisarjassa (Skrolli 2013.2–2014.2) luotua grafiikkamoottoria. Mikäli OpenGL-ohjelmointi ei ole ennestään tuttua tai kaipaat muistin virkistystä, kyseisten artikkelien lukeminen on suositeltavaa.

Stereoskooppisessa renderöinnissä on kaksi päävaihetta. Ensin renderöidään kummankin silmän näkymät tekstuureihin ja sen jälkeen yhdistetään ne ruudulla esitystä varten. Joissakin esitystavoissa vaiheet voidaan yhdistää ja renderöidä näkymät suoraan oikeaan paikkaan ruudulla, mutta Oculus Riftillä tämä ei ole mahdollista.

Jotta syvyysvaikutelma välittyisi käyttäjälle, näkymät on piirrettävä hieman eri kohdista. Katselupisteiden etäisyyden tulisi suunnilleen vastata katsojan silmien välistä etäisyyttä – muuten katsoja voi tuntea olonsa epämukavaksi.

LibOVR tarjoaa C-kielisen, oliopohjaisen rajapinnan. Kaikkien funktioiden, tietotyyppien ja vakioiden nimissä on etuliite ovr. Funktioiden nimistä ilmenee myös mihin oliotyyppiin ne liittyvät. Esimerkiksi ovrHmd-tietotyyppi kuvaa silmikkonäyttöä, ja siihen liittyvien funktioiden nimet ovat muotoa ovrHmd_DoSomething.

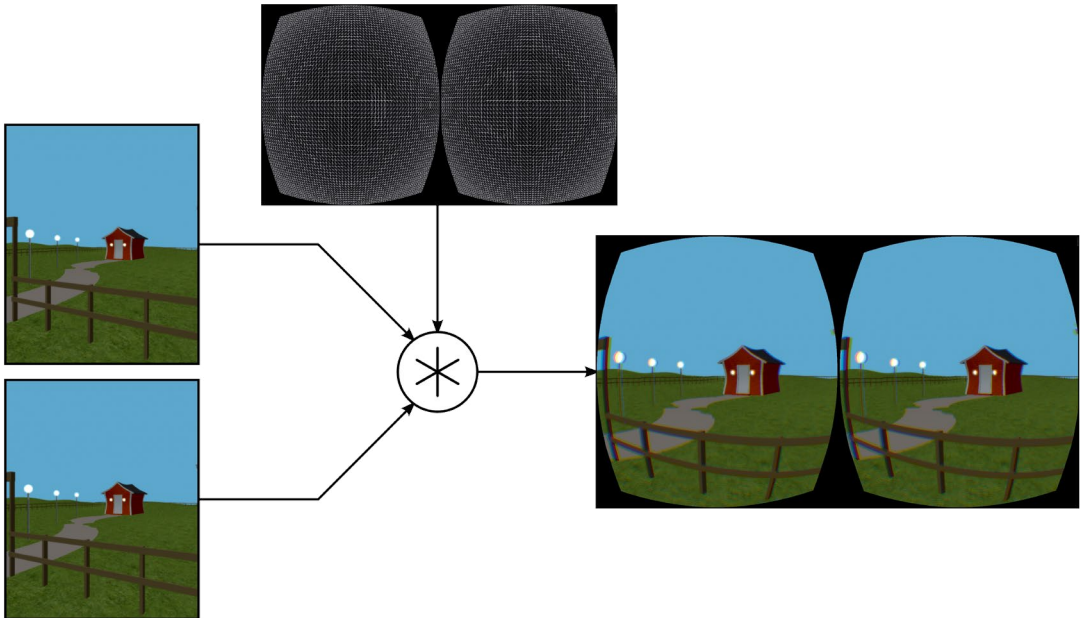
Ennen käyttöä kirjasto pitää alustaa kutsumalla ovr_Initialize-funktiota. Se palauttaa alustuksen onnistumista kuvaavan totuusarvon. Ohjelmistonkehityspakettiin sisältyvän taustaohjelman tulee olla käynnissä, tai alustus epäonnistuu. Linux-jakelut eivät ole vielä paketoineet LibOVR:ää, joten käynnistyksen joutuu virittämään itse.

Alustuksen jälkeen silmikkonäyttöä kuvaavan olion voi luoda funktiolla ovrHmd_Create. Funktio ottaa parametrina silmikkonäytön indeksin: arvo 0 tarkoittaa ensimmäistä järjestelmästä löytyvää silmikkonäyttöä. Jos funktio palauttaa null-osoittimen, tarkoittaa se, ettei yhtään silmikkonäyttöä löytynyt. Ohjelma voi tällöin vaihtaa tavalliselle näytölle sopivaan piirtotapaan. Listauksessa 1 on esitetty alustukseen tarvittavat toimet.

Oculus Riftin laaja kuvakulma perustuu näyttöpaneelin kuvaa vääristäviin linsseihin. Vääristymä on epälineaarinen siten, että lähempänä näkökentän keskipistettä pikselit ovat pienempiä ja tarkkuus suurempi. Jotta käyttäjä näkisi kuvan oikein, on ruudulle piirrettäessä käytettävä käänteistä vääristymää.

LibOVR:n ensimmäisissä versioissa vääristymä oli itse tuotettava shader-ohjelmalla ohjelmointioppassa esitettyjen kaavojen mukaan. Kaiken saaminen täsmälleen oikein oli melkoisen hankalaa. Versiosta 0.3.0 alkaen on tarjolla helpompi rajapinta, jossa kirjastolta voi pyytää valmiin kolmioverkon tekstuurikoordinaatteineen. Tämä tapahtuu ovrHmd_CreateDistortionMesh-funktiolla. Parametrina annetaan osoitin ovrDistortionMesh-olioon, johon funktio täyttää pyydetyt arvot. Funktiota täytyy kutsua erikseen kummallekin silmälle.

Valon eri aallonpituudet taittuvat linssissä hieman eri tavalla tuottaen erivärisiä reunuksia erityisesti näkökentän reuna-alueilla olevien esineiden ympärille. Näiden kromaattisten virheiden korjaamiseksi vääristymä-



Kummankin silmän kuvat renderöidään erikseen ja yhdistetään vääristymäverkon avulla lopulliseksi ruudulla näkyväksi kuvaksi.

verkon tekstuurikoordinaatit voi halutessaan pyytää erikseen eri värikomponenteille. Listaus 2 näyttää kuinka vääristymädata saadaan LibOVR:ltä ja kuinka sitä voi käyttää OpenGL:ssä.

Koska katsojan havaitsema pikselitiheys kulmayksikköä kohti vaihtelee, piirtoon tarvittavan näkymän koko ei vastaa näyttölaitteen erottelutarkkuutta. Jos kuva-ala halutaan hyödyntää kokonaan, tarvittava tarkkuus on hieman suurempi. Tässä tulee avuksi funktio ovrHmd_GetFovTextureSize, joka palauttaa parametrina annettavaa näkökenttää vastaavan tekstuurin koon.

Virtuaalitodellisuuteen uppoutumisen viimeistelee käyttäjän pään asennon seuranta. Se käynnistetään kutsumalla ovrHmd_ConfigureTracking-funktiota. Parametrina annetaan toivottuja sekä vaadittuja seurantaominaisuuksia kuvaavat bittimaskit. Funktion palauttama totuusarvo kertoo, onnistuiko seurannan käynnistys.

Seurannan ollessa käynnissä nykyisen tilan saa selville funktiolla ovrHmd_GetTrackingState. Paluuarvona saatava tietorakenne sisältää muunmuassa pään asennon ja sijainnin. Ne yhdistetään pelihahmon sijaintiin ja silmän siirrokseen lopullisen kameramatriisin muodostamiseksi.

Kirjaston alustuksen yhteydessä asennon seurannan nollakohta asetetaan noin metrin päähän seurantakameran eteen, katse kameraa kohti. Koska käyttäjän pää todennäköisesti ei ole lepoasennossa juuri tässä kohtaa ja käyttäjä saattaa myös vaihtaa asentoa pelaamisen aikana, on ohjelman syytä tarjota mahdollisuus nollaukseen nappia painamalla. Toiminto on helppo toteuttaa kutsumalla funktiota ovrHmd_RecenterPose.

Vaikka Oculus Rift tuottaakin varsin uskottavan vaikutelman kolmiulotteisesta ympäristöstä, aivan kaikkia ihmisen näkökyvyn ominaisuuksia se ei pysty jäljittelemään. Kaksiulotteisen näyttöpinnan kaikki pisteet ovat samalla etäisyydellä silmästä, joten koko kuva näkyy tarkkana. Syväterävyystehosteen käyttäminen silmikkonäytön kanssa ei ole kuitenkaan suositeltavaa, sillä nykyisillä laitteilla ei pysty seuraamaan käyttäjän katseen suuntaa.

Silmikkonäyttöä hyödyntävissä sovelluksissa kannattaa suosia peliohjaimen käyttöä, koska käyttäjä ei pysty näkemään käsiään ja näppäimistöä. Jos näppäimistön käyttö on nappien määrän tai hiiriohjauksen takia välttämätöntä, napit kannattaa sijoittaa lähelle toisiaan helposti löydettävään paikkaan. Useimmissa näppäimistöissä on F- ja J-näppäimissä kohoumat, joiden avulla ne löytää myös silmikkonäytön peittäessä näkökentän.

Pään asennon seurantaa käytettäessä pystysuuntaiset katselukontrollit kannattaa ottaa pois käytöstä ja käyttää yksinomaan asennon seurannasta saatavaa dataa.

Jotkin tehosteet, kuten varjokartat ja ympäristökarttoihin perustuvat heijastukset, vaativat apunäkymän piirtoa tekstuuriin ja sen käyttämistä osana varsinaisen näkymän piirtoa. Apunäkymää ei kannata turhaan piirtää erikseen eri silmille, vaan valmistelutoimet voi suorittaa yhteisesti. Tämä kannattaa ottaa huomioon grafiikkamoottorin suunnittelussa.

```
if(ovr_Initialize())
{
 ovrHmd hmd = ovrHmd_Create(0);
 if(hmd)
 {
  // Kaikki kunnossa
 }
 else
 {
  // Silmikkonäyttöä ei kytketty
 }
}
else
{
 // Alustus epäonnistui
}
```

Listaus 1: LibOVR:n alustus.

```
ovrDistortionMesh mesh;
ovrEyeType eye = ovrEye_Left; // Tai ovrEye_Right
ovrFovPort fov = hmd->DefaultEyeFov[eye]; // Oletusnäkökenttä on hyvä lähtökohta
ovrHmd_CreateDistortionMesh(hmd, eye, fov, ovrDistortionCap_Chromatic, &mesh);
ovrDistortionVertex &v0 = mesh.pVertexData[0];
glVertexAttribPointer(VERTEX, 2, GL_FLOAT, false, sizeof(ovrDistortionVertex),
        &v0.ScreenPosNDC.x);
glVertexAttribPointer(RED_TEXCOORD, 2, GL_FLOAT, false, sizeof(ovrDistortionVertex),
        &v0.TanEyeAnglesR.x);
glVertexAttribPointer(GREEN_TEXCOORD, 2, GL_FLOAT, false, sizeof(ovrDistortionVertex),
        &v0.TanEyeAnglesG.x);
glVertexAttribPointer(BLUE_TEXCOORD, 2, GL_FLOAT, false, sizeof(ovrDistortionVertex),
        &v0.TanEyeAnglesB.x);
// Muista kutsua ovrHmd_DestroyDistortionMesh(&mesh); kun dataa ei enää tarvita
```

Listaus 2: Vääristymäverkon pyytäminen LibOVR:ltä.

Oculus Rift DK2

Skrollin numerossa 2013.3 esittelimme Oculus Rift -silmikkonäytön. Viime syksynä laitteesta julkaistiin toinen kehitysversio. Mikä on muuttunut, ja onko suunta parempaan vai huonompaan?

Oculus Rift on vuonna 2012 joukkorahoituksen voimin luotu silmikkonäyttö, jolla voi uppoutua keinotodellisuuteen. Projekti lupasi päihittää silloin olemassa olevat laitteet ominaisuuksillaan ja samalla pitää hinnan kohtuullisena. Loppukeväästä 2013 saapunut ensimmäinen kehitysversio lunastikin lupaukset suurelta osin. Laite on sen jälkeen ollut harrastajien mukana monissa tietokonetapahtumissa kuten Assemblyssä. Syksyllä 2014 saatiin Developer Kit 2 eli DK2, johon tutustumme tässä artikkelissa.

DK2 on edeltäjäänsä huomattavasti kompaktimpi. Reilun tuuman verran pienemmän näyttöpaneelin ansiosta etureunan kolhosta ulkonemasta on päästy eroon. Ohjauselektroniikka on upotettu samoihin kuoriin näyttöpaneelin kanssa, joten erillistä sovitinlaatikkoa ei enää tarvita. Laitteesta lähtee HDMI- ja USB-kaapelit, jotka kulkevat pään yli niskaan ja sieltä tietokoneeseen. Kaapelit voi halutessaan irrottaa. Integroidun elektroniikan takia painoa on tullut hieman lisää.

Silmikon yläreunassa on USB-liitin, joka on kytketty integroituun USB-hubiin. Siihen voi kytkeä esimerkiksi langallisen peliohjaimen, tai virtuaalimaailman sisältävän USB-muistitikun.

Ehkä suurimpana uutuutena Rift pystyy nyt seuraamaan myös käyttäjänsä pään sijaintia. Tähän tarvitaan mukana toimitettavaa infrapunakameraa, joka asetetaan näytön päälle tai muuhun sopivaan paikkaan. Kameran seurantakeila on kolmisen metriä syvä ja yhtä leveä, mikä riittää paikallaan olevalle pelaajalle paremmin kuin hyvin. Kokonaan poispäin kamerasta ei kuitenkaan voi kääntyä. Seuranta toimii hyvin ja parantaa läsnäolon tunnetta erityisesti pieniä esineitä lähietäisyydeltä tutkiessa.

Näytön erottelukykyä on kasvatettu 1920×1080 pikseliin. Kumpikin silmä saa puolet näytöstä eli 960×1080 pikseliä. Suuri osa tästä kuluu kuitenkin näkökentän reuna-alueisiin, joten lopullinen tarkkuus tuo mieleen lähinnä VGA-aikakauden. Harmillisesti osapikselien täyttöaste ei ole juurikaan parantunut. Virkistystaajuutta on nostettu 75 kuvaan sekunnissa, joka yhdessä matalaviiveisen näyttöpaneelin kanssa tuottaa pään liikkeitä paremmin myötäilevän kuvan. Eron havaitseminen tosin on yksilöllistä ja riippuu myös käytetystä pelistä tai ohjelmasta.

Vaihdettavia linssejä toimitetaan aiemman kolmen parin sijaan nyt vain kahdet. Normaalinäköiset, kaukonäköiset ja lievästi likinäköiset pärjäävät samoilla linsseillä, ja ainoastaan voimakkaasti likinäköiset tarvitsevat erilliset linssit. Muille fysiologisille poikkeamille, kuten silmien etäisyydelle tai karsastukselle ei valitettavasti tarjota säätöjä.

Oculus Rift DK2:ta on vaikea arvostella koska sen hyvyys on kovin subjektiivista. Toiselle se on jännä lelu, josta jää lähinnä paha olo. Toiselle se on nyt ja jatkossa välttämätön lisävaruste josta maksaisi vaikka tuplahinnan. DK2 on tällä hetkellä paras myynnissä oleva kuluttajahintainen silmikko ja ainoa, jolle on peleissä tuki. Sen rauta on jo täysin kuluttajalaatuista, mutta softapuolen ongelmat ansaitsevat sille Developer Kit-nimen.

Jos pelaat pelejä joissa on Oculus Rift-tuki ja olet kiinnostunut virtuaalitodellisuudesta, kannattaa vahvasti harkita sen ostamsita. Muiden kannattaa kokeilla etukäteen. Vaikka kilpailevia tuotteita on luvattu, kestää aikansa ennen kuin peleihin toteutetaan tuki niille joten Oculus Riftin johtoasema on toistaiseksi vakaalla pohjalla.

Näkökenttä on hieman kutistunut aiemmasta, mutta muutosta ei käytännössä juuri huomaa. Näkymä on yhä riittävän laaja luodakseen uskottavan vaikutelman läsnäolosta. Musta reunus näkökentän äärirajoilla näyttää lähinnä siltä kuin kasvoilla olisi hiihtolasit.

Suuria pelijulkaisuja on saatu muutama, mutta valikoimalla ei vielä pääse juhlimaan. Sen sijaan indie-kehittäjät ovat ottaneet virtuaalitodellisuuden innolla vastaan. Oculus VR ylläpitää myös listaa johon kuka tahansa voi lähettää omia tuotoksiaan muiden kokeiltavaksi.

Laitteen kehityksessä on tapahtunut myös yllättävä poliittinen käänne: Facebook osti Oculuksen. Aika näyttää miten tämä vaikuttaa silmikkonäytön tulevaisuuteen. Useita muitakin yrityksiä on hypännyt keinotodellisuuskelkkaan mukaan ja kehittelee omia laitteitaan, joten laitetarjonnan voi odottaa laajenevan tulevaisuudessa.

Virtuaalitodellisuuslaitteita on vaikea arvioida yksiselitteisesti, koska todellisuuden kokeminen on lopulta subjektiivista. Joillekin se on paras juttu sitten Amigan, toisille se taas aiheuttaa lähinnä pahoinvointia. Rift DK2 on kuluttajien ulottuvilla olevista silmikkonäytöistä kuitenkin paras ja toistaiseksi ainoa, jolle on pelitukea. Jos siis haluat päästä virtuaalitodellisuuteen ensimmäisten joukossa, Oculus Riftiä voi suositella. Kilpailijoilla kestänee

pari vuotta kuroa etumatka umpeen.

Elite: Dangerous

E:D on malliesimerkki hyvin toteutetusta Oculus Rift-tuesta. Riftin käyttöönotto ei vaadi juurikaan säätämistä. Ohjaamojen ”hologrammi-hud” ja pelin valikot leijuvat pelaajan edessä eivätkä riko immersiota. Jotkut tekstit (esim chat) ovat hieman pienellä fontilla, mutta niistä saa kuitenkin selvää.

Avaruustaisteluissakaan OR:n resoluutio ei tuota ongelmia, sillä ne suoritetaan yleensä lähietäisyydeltä ja aluksen HUDin avulla toiset alukset näkyvät hyvin. Elitessä toiminta on keskimäärin kohtuu rauhallista, joten OR päässä olen istunut kolmekin tuntia yhteen menoon.

DCS: World

DCS:ssä OR:n resoluutio tulee vastaan monessa mielessä. Koneiden ohjaamot ovat aivan upean näköisiä ja immersio on huikea. Mutta resoluution vuoksi pienempiä mittareita ja HUD-symboleita ei näe riittävän tarkasti joten esimerkiksi vakavaan suihkukonekaluston lentämiseen OR ei kelpaa.

Itse olen lentänyt enimmäkseen lisäosana saatavalla Messerschmitt 109K-koneella, joka toisen maailmansodan koneena on suihkareita yksinkertaisempi tekniikaltaan ja mittareiltaan. Sen tärkeimmät mittarit ovat riittävän isoja, että ne näkee ja pienempiä voi kurkata nojaamalla eteenpäin. Ilmataistelut DCS:ssä ovat todella adrenaliininkatkuisia ja fiilis upea. Kutosen tarkistamiseksi täytyy oikeasti nojata sivulle ja kääntää päätä joten taisteleminen on fyysisesti yllättävän raskasta. Todellisuudessa olisi lisänä vielä G-voimat, joten ei ole ihme että ilmataistelut olivat lyhyitä. OR:llä yhden muutaman minuutin ilmataistelutehtävän jälkeen on niin poikki, että täytyy pitää taukoa ja pyyhkiä linssit hiestä. Valitettavasti DCS:stä puuttuu vielä kunnon kampanja toisen maailmansodan kalustolla, joten sen lentäminen on jäänyt kohtuu vähäiseksi.

Verkossa pelattaessa OR:n resoluutio aiheuttaa jälleen ongelmia. Pelissä on pakko pitää ikonit (toiset koneet näyttävät tekstit) päällä, tai muuten ei taivaalta löydä ketään, ennenkuin on liian myöhäistä. Useimmilla palvelimilla ikonit on pakotettu pois käytöstä joten niillä on turha koittaa pelata OR:n kanssa.

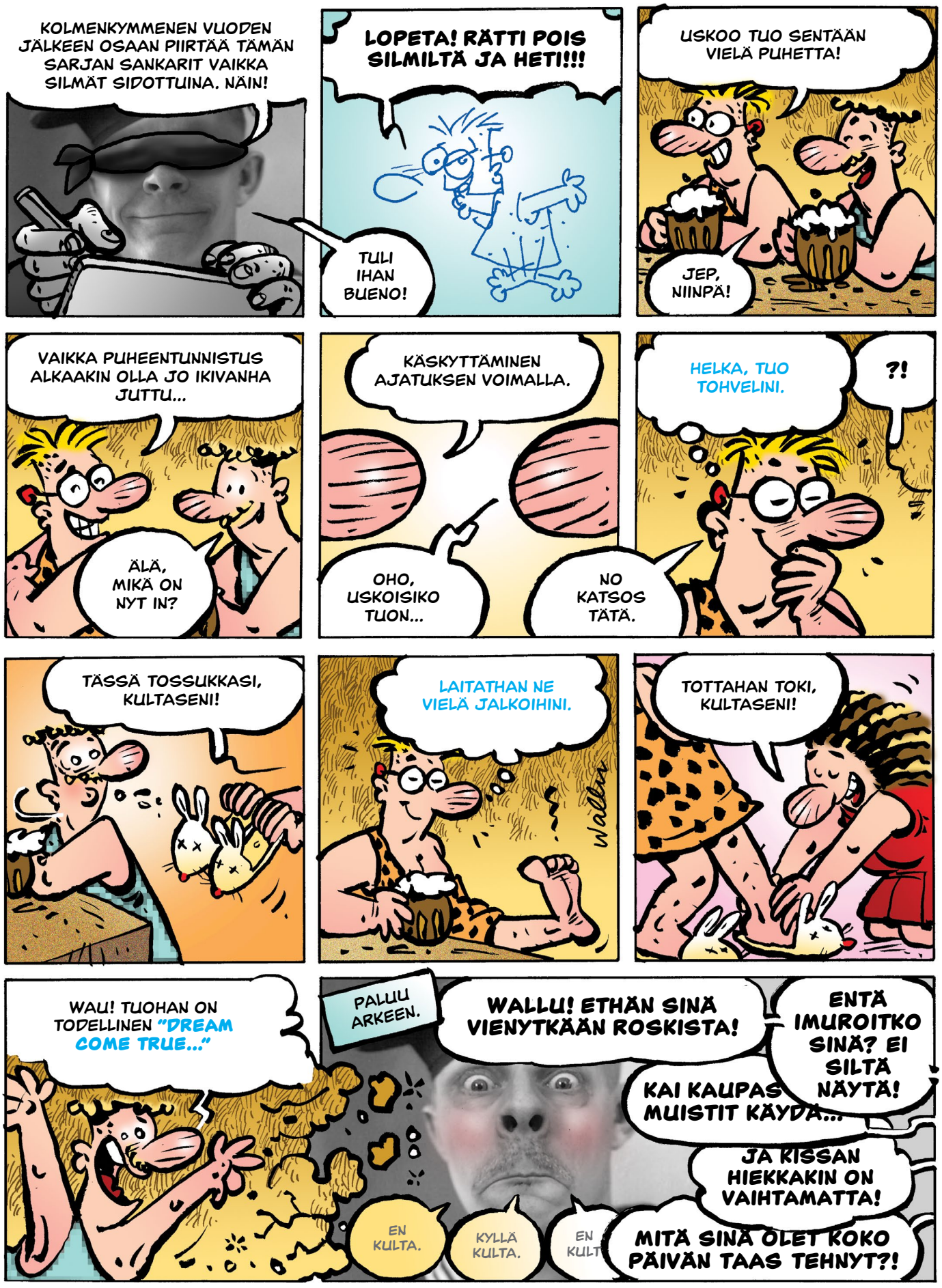
DCS:n valikot eivät ole tehty OR:ää varten, vaan ne ovat kaksiulotteiset. Päävalikko ei näy OR:n näytöllä ollenkaan. Tehtävän debriefing-näkymä taas näkyy, mutta siitä ei tahdo saada selvää.

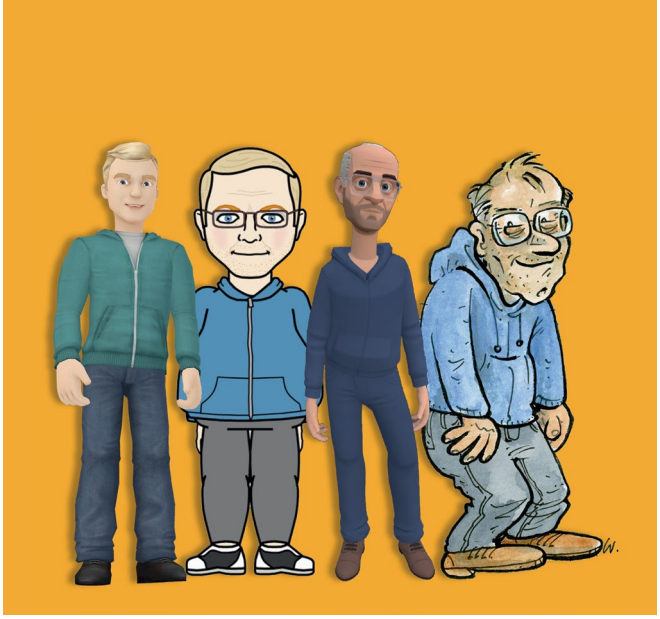
War Thunder

War thunderissa on ihan toimiva OR-tuki. Ohjaamot eivät kuitenkaan ole jollain tapaa yhtä hienoja ja teräviä kuin DCS:ssä. Pelistä löytyy realistinen moodi, jossa on ikonit päällä mutta lentomallit ovat realistisia. Valitettavasti pelejä käydään ilmeisesti kohtuu vähän, joten varttitunnin odottaminenkin on yleistä.

Half Life 2

Half Life 2:n betassa on virtual reality-moodi, jonka avulla pelin pitäisi toimia OR:llä. Tätä moodia en saanut toimimaan millään tempuilla. Kuva tulee kyllä OR:lle, mutta valikon lisäksi kaikki on mustaa. Internetissä moni muukin on valitellut samaa. Osa on saanut toimimaan vaihtamalla OR:n ykkösnäytöksi, minä en. Tämä on hyvä esimerkki siitä, että softapuolella on vielä paljon tekemistä.


KOLMENKYMMENEN VUODEN JÄLKEEN OSAAN PIIRTÄÄ TÄMÄN SARJAN SANKARIT VAIKKA SILMÄT SIDOTTUINA. NÄIN!
LOPETA! RÄTTI POIS SILMILTÄ JA HETI!!!
TULI IHAN BUENO!
USKOO TUO SENTÄÄN VIELÄ PUHETTA!
JEP, NIINPÄ!
VAIKKA PUHEENTUNNISTUS ALKAAKIN OLLA JO IKIVANHA JUTTU...
ÄLÄ, MIKÄ ON NYT IN?
KÄSKYTTÄMINEN AJATUKSEN VOIMALLA.
OHO, USKOISIKO TUON...
NO KATSOS TÄTÄ.
HELKA, TUO TOHVELINI.
?!
TÄSSÄ TOSSUKKASI, KULTASENI!
LAITATHAN NE VIELÄ JALKOIHINI.
Wallu
TOTTAHAN TOKI, KULTASENI!
WAU! TUOHAN ON TODELLINEN "DREAM COME TRUE..."
PALUU ARKEEN.
WALLU! ETHÄN SINÄ VIENYTKÄÄN ROSKISTA!
ENTÄ IMUROITKO SINÄ? EI SILTÄ NÄYTÄ!
KAI KAUPAS MUISTIT KÄYDÄ...
JA KISSAN HIEKKAKIN ON VAIHTAMATTA!
MITÄ SINÄ OLET KOKO PÄIVÄN TAAS TEHNYT?!
EN KULTA.
KYLLÄ KULTA.
EN KULT



Kivikautinen sarjakuva

Mikrokivikausi-sarjakuva alkoi ilmestyä niihin aikoihin, kun isoisä ensimmäisen tietokoneen osti. Siis 30 vuotta sitten.

Teksti: Ulla W

30 vuotta on pitkä aika duunissa kuin duunissa. Mutta mikä ettei, jos työstään tykkää. Sarjakuvataiteilija Wallu on jaksanut tehdä Mikrokivikausi-sarjaansa vuodesta 1985. Eli ainakin tekijä on pitänyt sarjastaan. Ja 30 vuotta sitten siitä piti joku toinenkin. M.O.T

Paluu alkuun

Sarjakuvahulluus iski Walluun varhain. Vuoteen 1985 mennessä hän oli jo ehtinyt voittaa Kemin sarjakuvakilpailut kahdesti, kustantaa viisi omakustannesarjakuvalehteä, piirtää poliittisia pilapiirroksia Ilta-Sanomiin sekä kuvittaa Tecnopressin ja sittemmin Sanomien kustantamia Prosessori-, Tietokone- ja Mikrobitti-lehtiä. Prosessorissa ilmestyi monisivuinen ”Missä vika, kun projekti meni pieleen?” -opetussarjakuvakin.

Mikrobitin alkuvaiheissa Wallu piirsi kuvituskuvia ohjelmatulosteitten yhteyteen. Luolastoja, lohikäärmeitä ja muuta mielenkiintoista. Vaan yksi puuttui: lehdessä ei ollut sarjakuvaa. Eikä sitä sinne haluttukaan. Lehti halusi tarjota lukijoilleen hyödyllistä (lue: myyvää) asiatietoa, eikä sarjakuvasta ollut sellaiseen, ajateltiin.

Sinnikäs taiteilija ei luovuttanut. Hän suunnitteli ja piirsi malliksi kuusi puolen sivun kokoista, mustavalkoista sarjakuvaa (Mikrokivikaudet 1–6) ja vei ne innoissaan toimitukseen. Siellä ne tyrmättiin: veivät lehdestä liikaa tilaa.

Kun huhtikuun 1985 lehti oli jo painossa, onnekas sattuma puuttui peliin. Yksi puolen sivun ilmoitus peruuntui aivan viime hetkillä. Normaalisti tällaiset aukot täytetään kustantajan omilla ilmoituksilla, mutta syystä tai toisesta nyt tilalle laitettiin ensimmäinen Mikrokivikausi-sarjakuva. Säälistä, uskoo taiteilija.

Lehti ilmestyi ja lukijoiden palaute sarjakuvasta oli niin myönteistä, että sitä päätettiin julkaista lisää, jos lehdessä vain oli tilaa. Onnekkaasti sarja mahtui myös toukokuun numeroon. Se ilmestyi siitä lähtien jokaisessa Mikrobitin numerossa aina vuoden 2013 syyskuuhun, jolloin se irtisanottiin tietotekniikan kehittymisen mahdollistamalla persoonattomalla sähköpostiviestillä: ”Lehden palstojen uudelleenjärjestelyssä ja sisällön karsimisessa tulimme siihen tulokseen, että emme enää jatka Mikrokivikautta uudessa MB:ssä.” Ja Pimeys valtasi piirtäjän mielen.

Onneksi hätiin tuli vuoden 2014 alusta Skrolli, joka antoi sarjalle ja taiteilijalle uuden elämän.

Matkan varrelta

Palataan vielä vuoteen 1985. Syksyn tullen Wallu siirtyi opettajan virasta vapaaksi taiteilijaksi ja samalla Mikrobitin freelance-graafikoksi ja -taittajaksi. Seuraavien seitsemän lihavan vuoden aikana Mikrokivikausi kukoisti: vain vuosi ensijulkaisusta ja sarja sai jo värit. Vuosi tästä ja ensimmäinen koko sivun kokoinen sarjakuva ilmestyi.

Vuoden 1987 ensimmäisessä sarjassa oli ruutu, jonka Wallu oli omakätisesti Amigalla piirtänyt. Kyseinen piirros piti tuohon aikaan valokuvata tietokoneen ruudulta. Sen jälkeen kuva lähetettiin lehteen tulevan kuvamateriaalin mukana kirjapainoon, jossa se skannattiin ja sijoitettiin tyhjään sarjakuvaruutuun. Huimaa!

Myös piirrosteknisesti sarja kehittyi harppauksin. Ensimmäiset sarjat Wallu piirsi mustalla huopakynällä A4-kokoiselle monistuspaperille. Kokeiltuaan väritystä kirjapainon tasaväreillä (ja petyttyään pahasti) taiteilija tarttui vesiväreihin ja sen ajan hittivärittimeen eli retussiruiskuun ja teki seuraavat sarjat niillä. Lopputulos tyydytti taiteilijaa, ja tästä tuli pitkäksi aikaa sarjan vakiototeutustapa. Lisäksi sarja siirrettiin piirustuspaperilta lehteen Sanomien huippuskannereilla ja ruiskuväritkin toistuivat hienosti.

Sitten tuli 90-luku, lama ja kotiskannerit. Wallullekin tuli käsky siirtää sarjansa digitaaliseen muotoon kotikonstein. Agfan skannerilla se kävi, mutta ilman minkään valtakunnan skannauskoulutusta jälki ei aina ollut parasta A-luokkaa. Tästä kuitenkin seurasi vääjäämättä se, että taiteilija alkoi käyttää enemmän tietokonetta sarjan tekemisessä, etenkin sen värittämisessä. Silti vielä tänäänkään Mikrokivikausi ei synny kokonaan tietokoneella. Wanhuksen on pakko piirtää käsin sivellinkynällä, kun on tuon taidon vuosien varrella näppeihinsä saanut. Suotakoon se hänelle.

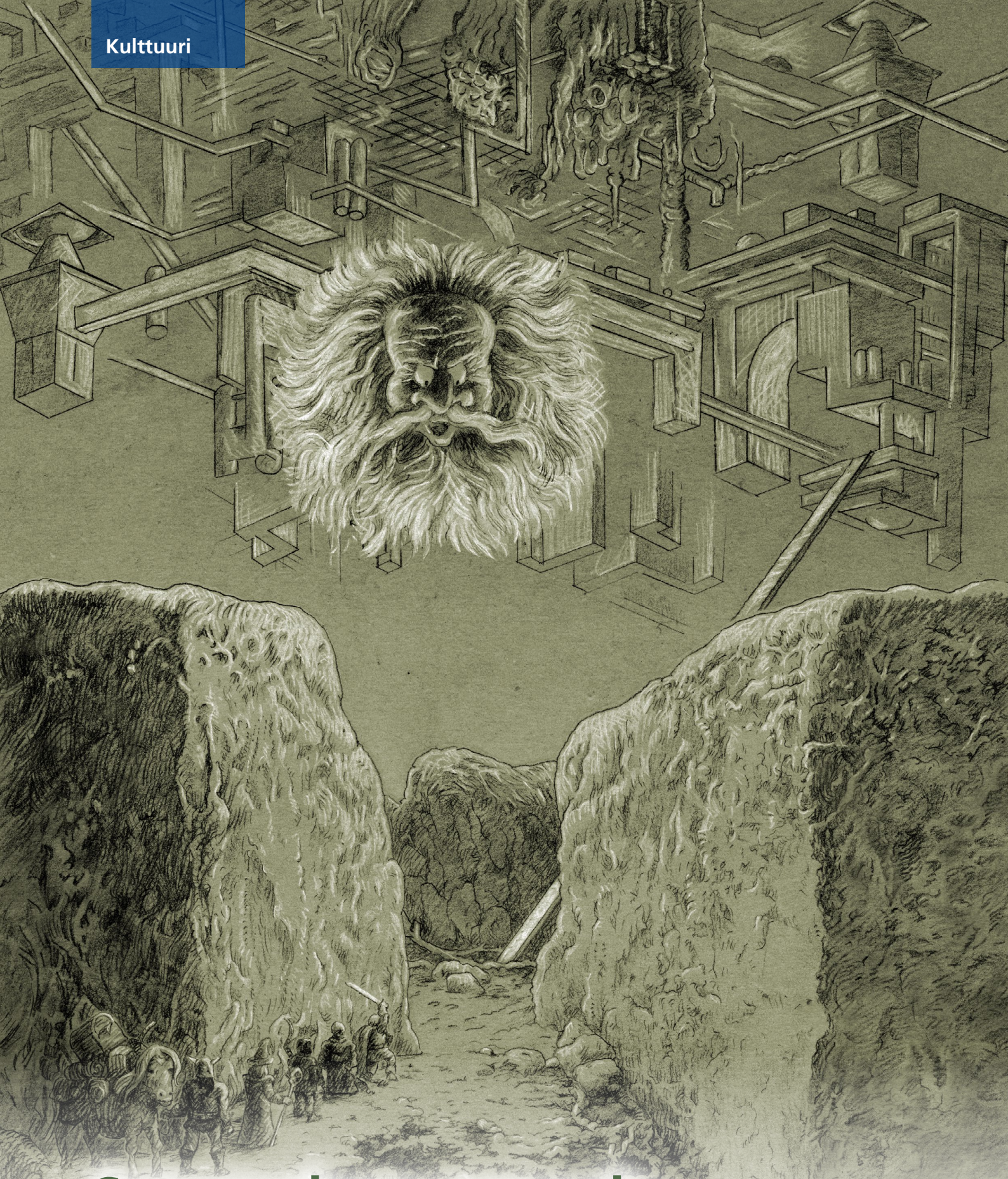
Juhlaa ja arkea

Osoitteessa mikrokivikausi.fi on Wallun itsensä ylläpitämä arkisto kaikista Mikrobitissä julkaistuista sarjoista. Komeimmat hetket ovat kenties sarjan alkutaipaleelta, jolloin sarja sai lehdestä parhaimmillaan kolmekin sivua tilaa: keskiaukeaman 3D-juliste ja sivun sarja sekä maininta tästä kuvan kera jo kannessa.

Myös erilaisia tasavuosia on aina juhlittu näyttävästi, alkaen 5-vuotisjuhlista vuonna 1990, jolloin sarja oli lehdessä keskiaukeaman kokoisena julisteena. Paljastavin ja taiteilijan itsensä mielestä kenties onnistunein juhlasarjis on vuodelta 2005, jolloin sarja täytti 20 vuotta. Silloin kaivettiin jo hieman hapantunut tekijäkin esiin.

Mikrokivikausi on ollut vuosien varrella muutaman kerran mukana näyttelyissä niin Suomessa kuin ulkomaillakin. Viime vuoden lopulla yksi sivu matkasi kiinaksi googlekäännettynä Taiwaniin, jossa se oli komeasti esillä paikallisessa sarjakuvatapahtumassa (The 15th International Comic Artist Conference, Kaohsiung, Taiwan). Nyt Wallu tietysti odottelee siltä suunnalta kustantajan kutsua. Wallulle myönnettiin viime vuonna Suomen sarjakuvaseuran sarjakuvaneuvos-arvonimi, joten ihan hukkaan nämä 30 vuotta Mikrokivikautta eivät ole hänen osaltaan menneet.

Onnea 30-vuotiaalle!



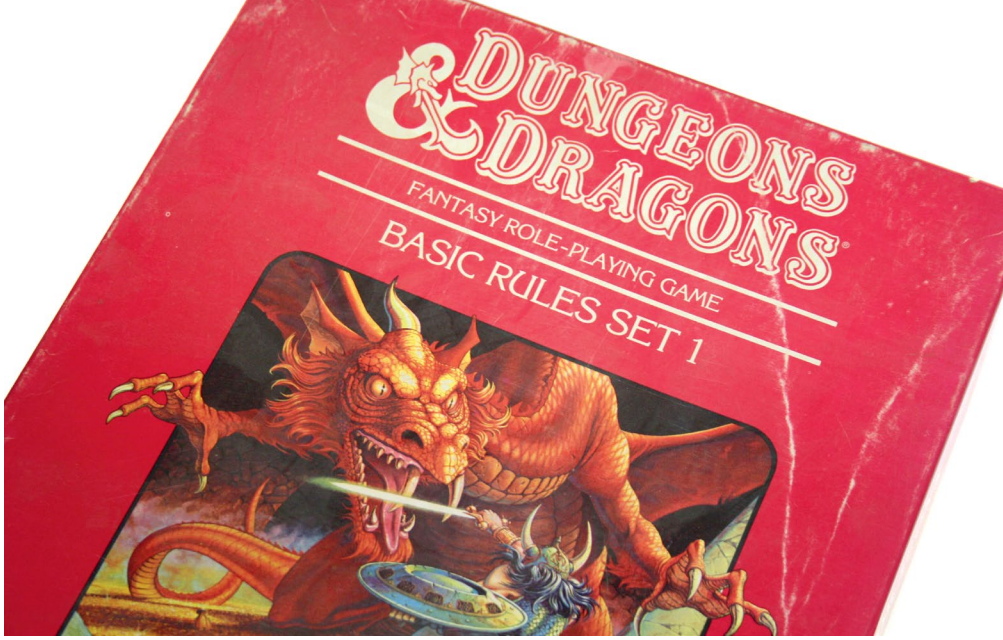
Satunnainen samooja

Kolme vuosikymmentä suomalaista roolipelaamista

Suomalaisten roolipelien 30-vuotista historiaa valaisevan juttusarjan ensimmäisessä osassa veteraani muistelee läpi ensimmäiset viisitoista vuotta omia ropelluksiaan.

Teksti: Nordic the Incurable

Kuvat: Mikko Torvinen, Nordic the Incurable, Jukka O. Kauppinen



Olen ollut kiinnostunut peleistä niin kauan kuin niistä mitään olen ymmärtänyt. Muistan jo alle kouluikäisenä rakennelleeni legopalikoista erilaisia kuularatoja ja autojen takaa-ajopelejä. Seuraavaksi kävin läpi peruspelit Monopoleineen, Kimbleineen ja jopa Afrikan tähtineen – josta en, monen muun pelin ohella, oikein pitänyt sen liian suuren satunnaisuuden vuoksi

Tulin tuolloin pelanneeksi monet nyt jo unhoon jääneet lautapelit. Kaikki kelpasi ainakin kokeiltavaksi. Pelitarjonta ei ollut mitenkään huikaisevaa, ja kovin monet olivat perinteisen ”heitä noppaa ja liiku ruutuja” -tyylin edustajia, joita ei Erkkikään kauaa jaksanut pelata. Niin mukavasti kuin isä muuten pelaamiseeni suhtautuikin.

Omia pelejä suunnittelin lähes heti, kun kynä alkoi pysyä kädessä. Usein tein mielestäni parempia ja monipuolisempia versioita olemassa olevista peleistä. Joskus myös toteutin omalta tuntuvia ideoita. Hyvin ne tuntuivat toimivan. Kavereiden kanssa pelasimme paljon enemmän itse tehtyjä kuin kaupasta ostettuja pelejä.

Digitaalimutka

Roolipelit, nuo ihmismieltä viekkaasti haltioittavat tekeleet, ilmaantuivat elämääni mutkan kautta. 1980-luvun alussa pelailin paljon silloista ykkösluokkaa edustavalla Intellivision-pelikonsolilla, jolle oli peli nimeltä Advanced Dungeons & Dragons. Siinä ohjattiin luolissa könyävää seikkailijaa, joka vain jousi ja nuolet apunaan etsi aarteita ja ulospääsyä hirviöiden vartioimasta luolastosta – joutuakseen aina vain seuraavaan luolastoon. Jossakin odotti se loppuluolastokin, mutta sinne ei päässyt ihan joka toinenkaan kerta. Pelin mainoksessa ollut maininta ”perustuu suosittuun seurapeliin” oli herättänyt mielenkiintoni. Mikä ihmeen peli se sellainen oli, jossa saattoi vapaasti seikkailla luolissa, etsiä aarteita ja taistella hirviöitä vastaan? Mitään kuvausta muistuttavaa ei ollut vielä eteeni osunut.

Seuraavana vuonna ilmestyi vielä hienompi Advanced Dungeons & Dragons: Treasure of Tarmin, jossa hienosti ”kolmiulotteisia” luolastoja katseltiin seikkailijan silmin. Vaikeimmalla tasolla peli vei aikaa useita tunteja eikä loppuun pääseminen ollut kuin ohut toive. Näitä pelejä kavereiden kanssa pelatessa kului muutamakin sata tuntia ja kiinnostus ”sitä oikeaa peliä” kohtaan sen kuin kasvoi. Koska nettiä ei ollut eivätkä tietosanakirjat tai suomalaiset kaupat asiaa tunteneet, oli vain odoteltava, josko jostain ilmaantuisi tietoa tästä ihmeellisestä seikkailupelistä. Noita aikoja muistellessa ei voi kuin haikeana huokaista: kuinka helppoa onkaan nykyään etsiä ja saada tietoa kiinnostuksensa kohteista!

Kohtuullisen kärsimättömänä ja toimeliaana henkilönä en jäänyt odottamaan ihmepelin inkarnoitumista, vaan aloin suunnittelemaan omaa versiota. Ensin tein useita lautapelityyppisiä pelejä. Niissä seikkailija tai useampi kulki luolastoissa tarvikkeita ja aarteita etsien ja vastaantulevat monsterit murskaten. Näiden pelien upein idea oli reitin vapaa valinta, ainakin joiltakin osin. Pelaajat saattoivat kerätä voimia, aseita ja tarvikkeita kiertämällä helpommissa luolissa ennen kuin uskaltautuivat isompien hirviöiden eteen.

Sittemmin suunnittelin yhden pelaajan seikkailun, jossa kirja toimi ikään kuin tietokoneen roolissa, tavallaan pelinjohtajana. Sen sain melkein julkaistua, mutta kustantajan mukaan aika ei ollut vielä kypsä moisille ihmeellisyyksille. Muutama vuosi myöhemmin markkinoille putkahti englannista käännettyjä soolopelikirjoja, jotka oli tehty aivan oman ideani nuotilla.

Amerikan malliin

Ensimmäinen oikea roolipelini, Aciremа, hahmottui lopulliseen muotoonsa 80-luvun alkuvuosina. Siinä oli jo hyvinkin oikeat rope-elementit hahmoineen, pelinjohtajineen ja taistelusääntöineen. Kaikesta kyllä huomasi, että peli oli paljon velkaa pelaamilleni AD&D-videopeleille. Esimerkiksi kääpiöt olivat hirviölaji, eivätkä suinkaan hahmoluokka pelaajille. Tämä johtui tietysti siitä, että Tarminissa yksi perusviholisista oli vihainen kääpiöhahmo. Samaten hyvin oleellista oli se, että aseita ja varusteita ei ollut paljoa mukana seikkailuun lähdettäessä. Niitä löytyi vasta luolien uumenista, ties kenen sinne jättäminä.

Aciremaa suunnitellessani yksi tärkeä pointti oli se, että pelin oli oltava riittävän helppo ja selkeä opetella. Halusin antaa muillekin kiinnostuneille mahdollisuuden tutustua roolipeleihin – niitä kun ei Suomessa ollut vielä lainkaan myytävänä. Myin peliä (kaksi nidottua valokopiovihkoa) postitse ja markkinoin sitä mainitsemalla sen muutamaan kertaan Mikrobittilehdessä vuonna 1984 alkaneella palstallani. Päätoimittaja hyväksyi kaupallisuuden vivahteen, kun se ei ollut keneltäkään muulta pois.

Kuka Nordic?

Suomalaisen roolipelimaailman grand old man Nordic the Incurable on tehnyt ja julkaissut vuosikymmenten mittaan kymmeniä roolipelejä, pelannut läpi valtavan määrän muiden luomuksia ja kirjoittanut roolipeleistä viiteen eri lehteen. Tietokonepelaajat muistavat Nordicin Mikrobitti-lehdessä ilmestyneestä Peliluola-palstasta, jonka kautta Nordic loi Suomeen kukoistavan pohjan sekä tietokone- että perinteisille roolipelaajille. Nordic työskentelee yhä pelialalla, tekee edelleen roolipelejä ja pelaa viikoittain.

Joskus 80-luvun puolessavälissä satuin sitten lukemaan Aamulehdestä ison jutun, joka kertoi mielikuvituksellisesta harrastuksesta: roolipelaamisesta! Siinä vantaalainen Lauri Tudeer esitteli harrastustaan ja selvisi, että häneltä voi postitse tilata D&D-roolipeliä. No jopas ampaisin pääpostiin etsimään Helsingin puhelinluetteloa ja sieltä Lauri Tudeerin nimeä. Ei muuta kuin soitto herralle ja D&D tilaukseen.

Vielä vuosia tuon tapahtuman jälkeen ainoat kanavat, joista kotimaassa sai tietoa roolipelaamisesta, olivat Lauri Tudeerin postimyyntiluettelo, Conan-lehden roolipelipalsta ja Mikrobitin Peliluola, johon joskus kirjoitin jotain ropeasiaa. Kyllä niitä sitten kuulemma luettiinkin.

Nämä mainitsemani asiat olivat oleellinen osa sitä tapahtumaketjua, joka teki minusta roolipelaajan ja pelintekijän ja josta löytyi harrastus ja ammatti kymmeniksi vuosiksi.

Roolipelimarkkinat

Aciremaa myin ensin itse postitse. Myöhemmin, kun Lauri Tudeer avasi Fantasiapelit-liikkeen, hän otti sen myyntiin. Aciremaa myytiin kaikkiaan joitakin satoja kappaleita. Se oli jälkeenpäin katsottuna aika hassu pieni peli, mutta antoi varmasti monelle aloittelijalle sysäyksen roolipelaamiseen maailmaan. Mainittakoon myös, että viime vuonna Acirema käännettiin englanniksi, kuvitettiin hienosti ja julkaistiin netissä kenen tahansa ladattavaksi.

Seuraava pelini oli Miekka ja magia. Se perustui kokemuksiini Aciremasta, mutta oli ulkoasultaan jo hienompi ja oikein painotyötä. Se mainitaan usein ”ensimmäisenä suomalaisena roolipelinä”, eikä tuo arvio poikkea kauas totuudesta. Miekkaa ja magiaa myytiin noin tuhat kappaletta, joka on vielä nykyäänkin hyvä myyntimäärä suomalaiselle ropelle. Se on myös ansainnut paikan reilun vuoden kuluttua avattavassa Suomen pelimuseossa Tampereella.

Toki muutkin tekivät noihin aikoihin omia roolipeliviritelmiään. Muistan ainakin sellaisen pelin kuin Nousius, mutta tekijöitä en tunne. Kahdeksankymmentäluvun lopussa syntyivät Suomen ensimmäiset roolipelikaupat, joiden myötä markkinoille alkoi tulla monelta taholta suomenkielisiä roolipelejä, sekä kotimaisia että käännettyjä.

Nelostuote julkaisi Pasi Janhusen suunnitteleman Ankhin, Protocol Productions käänsi Dungeons & Dragonsin ja Ace-pelit kustansi suomenkielisen Runequestin. Myös pienjulkaisija Dada-pelit oli mukana suomalaisella materiaalilla. Seuraavina vuosina roolipelilehdet Seikkailija, Sininen lohikäärme ja Magus ilmaantuivat jakamaan ropetietoutta uusille harrastajille. Yhtäkkiä tavaraa oli tarjolla ylenpalttisesti ja ihan selväksi suomeksi. Englanninkielistä tarjontaa oli vielä monin verroin enemmän, joten ropeltajille alkoivat varsin hyvät ajat.

Vapaus vs. Sääntökirja

Ensimmäinen ostopelini oli Dungeons & Dragons Basic Set, kuuluisa punalaatikko. Sittemmin se ilmestyi samannäköisenä myös suomeksi. Peli kuuluu niihin varhaisiin ja upeisiin muistoihin, jotka liittyvät nimenomaan tutustumiseen, kirjojen kuvitukseen ja siihen uskomattoman hienoon tunnelmaan, joka outoon maailmaan syventyessä syntyi. Pelinä se ei ollut kummoinen, enkä itse asiassa koskaan edes pelannut tuota ensimmäistä D&D-seikkailuani.

Vaikka olin suunnitellut sääntöjä kymmeniin jollen satoihin eri peleihin ja ohjelmoinut Commodore 64:lle pitkiä seikkailupelejä, pidin D&D:n sääntöjä vaikeina ja monimutkaisina. Pelin kankeus ja sääntöjen paljous hämmensivät minua, koska olin tottunut omissa peleissäni paljon vapaampaan kulkuun. Ajattelin, että ilmeisesti jenkeille kaiken täytyi olla valmiiksi säänneltyä ja että heillä ei laskettu kovin paljon oman mielikuvituksen käytön varaan.

Tavallaan olin tai olen oikeassa, koska ajattelen edelleen hiukan samalla tavalla. Ymmärrän kyllä mainiosti, että jotkut (itse asiassa yllättävän monet) haluavat roolipeliltä tarkat ja selvät säännöt eri asioista. Se pitää asiat ymmärrettävinä ja antaa tietynlaisen turvan sekä pelaajille että pelinjohtajalle. On lohduttavaa, että on olemassa Sääntökirja, johon voi tarvittaessa tukeutua.

Minusta on kuitenkin palkitsevaa pelata suhteellisen vapaalla tavalla. Pelinjohtajana pidän maailman avoinna kaikille pelaajien ideoille ja olen valmis säveltämään asioita tilanteen mukaan. Pelaajana arvostan toiminnan ja ajatusten vapautta sen sijaan, että tuntisin kulkevani tarkoin rajatussa putkessa. Tämä ei tarkoita, että maailma muuttuisi jatkuvasti tai että kuka tahansa voi ottaa tarinan ohjat. Kannatan kyllä pelinjohtajan rajatonta valtaa, mutta edellytän häneltä riittävää älyä ja halua käyttää valtaansa fiksusti – koko seikkailun ja porukan eduksi.

Varsinkin pelaamisen alkuaikoina törmäsin usein käsitykseen, että roolipelissä olisi jonkinlainen pelinjohtaja vastaan pelaajat -asetelma. Jotkut pelasivatkin noin, iloiten pelinjohtajana siitä, että olivat onnistuneet murskaamaan hahmot mahdollisimman ovelilla keinoilla. Se tuntui oudolta tavalta suhtautua pelaamiseen. Mielestäni tärkeintä on hyvä tarina, hienot kokemukset ja jännittävä peli eikä suinkaan se, noudatetaanko sääntöjä pilkuntarkasti tai kuka voittaa kenet.

D&D:n lisäksi tutuiksi tulivat myös ANKH ja Runequest. ANKHista haluaisin pystyä sanomaan enemmänkin hyvää, mutta kaupalliseksi tuotteeksi se oli keskinkertainen. Pahin pettymys oli sen sääntöjen nojaaminen liikaa D&D:iin. Positiivista oli suomalaisen mytologian mukanaolo, viittasihan lyhentämätön nimikin Kalevalaan.

Runequest taas oli äärettömän monipuolinen ja monimutkainen peli varsinkin aloittelijan näkökulmasta. Ja aloittelijoitahan suurin osa ropeltajista tuolloin oli. Peli tarjosi toimivan sääntökokoelman ja erittäin mielenkiintoisen, näennäisrealistisen maailman. Sen pelaaminen oli kuitenkin aikamoista ankkain taaperrusta, ja sankariksi nouseminen jäi monella ikuiseksi haaveeksi.

Sekä D&D:iin että Runequestiin julkaistiin mukavasti lisämateriaalia seikkailujen ja kampanjoiden muodossa, mikä helpotti valmista materiaalia kaipaavan pelinjohtajan työtä.

1990-luku toi suomipelien tulvan

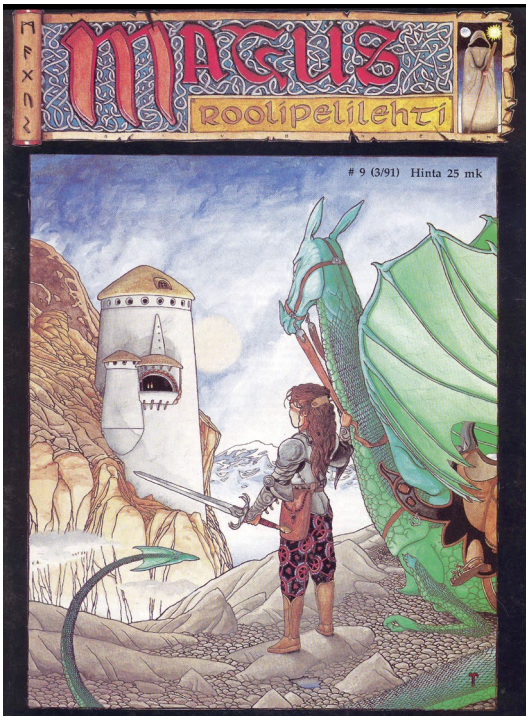
1990-luvun alussa suomenkielisiä pelejä julkaistiin lisää. Markkinoille tulivat muun muassa Tie tähtiin 2300, Twilight 2000, Keski-Maa ja Cyberpunk. Hiukan myöhemmin seurasivat Paranoia, Cthulhun kutsu ja Taru sormusten herrasta. Itse innostuin Keski-Maan tapahtumien siivittämänä haltioiden elämästä niin paljon, että väkersin haltiaroolipeli Rapierin, joka myytiin loppuun. Menestyksen siivittämänä tein uuden version nimellä Elhendi. Näissä oli kyse haltiakeskeisestä seikkailusta fantasiamaailmassa kepeähköjen mutta toimivien sääntöjen avittamana. Itse pidin erityisesti Elhendin taistelusysteemistä ”vaarallisine osumineen”.

90-luvun alussa tein myös ainoan kauhuroolipelini Astran, joka muistutti osin vähän liikaa ilmeisenä esikuvanaan toiminutta Call of Cthulhua. Hassua niin, sillä peliä tehdessäni koetin erityisesti välttää liiallista samankaltaisuutta. Lohduttauduin sillä ajatuksella, että kukapa tässä nyt ihan uusia kauhuaiheita keksisi, kun kaikkihan on jo jossain muodossa julkaistu. Astrasta ei tullut suurta hittiä.

Cyberpunk oli maailmaltaan mahtavan uusi ja erilainen. Se tempaisi vastustamattomasti mukaansa teknisellä ja silloin futuristisella menollaan. Sääntösysteemissä tosin oli muutamia heikkouksia: esimerkiksi hahmonluonti ei ollut viimeisen päälle suunniteltu. Mutta kuten aina ennenkin, muutamia omia muutoksia tekemällä siitä tuli varsin suosittu porukkamme keskuudessa.

Suomalaiset roolipelilehdet kilpailivat pienistä markkinoista aluksi tehokkaasti. Aika pian Sininen lohikäärme kuitenkin kuihtui pois, ja Seikkailijan ura taisi jäädä noin kymmeneen numeroon. Magus jatkoi ilmestymistä aina numeroon 50 asti. Itse osallistuin Sinisen lohharin ja Maguksen tekemiseen. Seikkailijan maailma Timo Konttisen johtamana tuntui liian etäiseltä minulle.

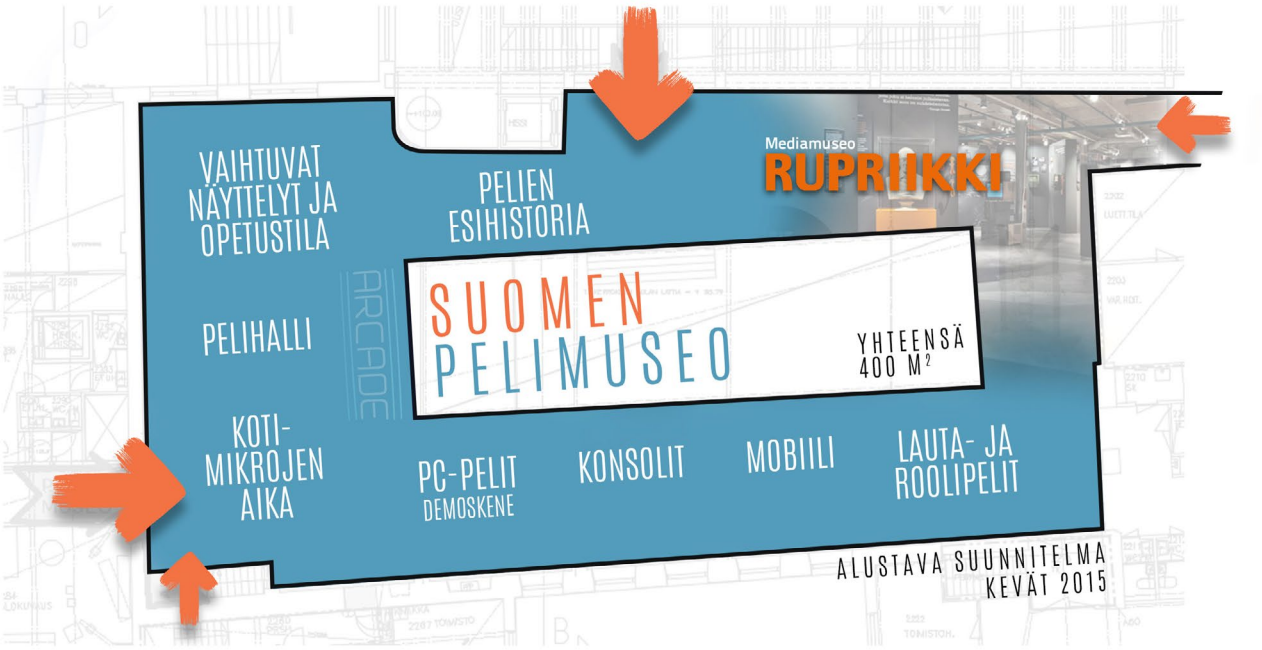
Myöhemmin tein myös omaa Claymore-lehteämme 20 numeron verran. Se oli hauskaa, vaan ei sillä rahoiksi



lyöty. Sittemmin Suomen ropelehtimarkkinoilla onkin ollut hiljaista lukuun ottamatta Roolipelaajaa, joka jaksoi yrittää usean vuoden ajan nousematta ilmeisesti koskaan kovin kannattavaksi.

Acirema ladattavissa osoitteesta http://www.roguelantern.com/blog/images/acirema-archive.zip.

Nordicin ”kolme vuosikymmentä suomalaista roolipelaamista” -reportaasi jatkuu seuraavassa Skrollin numerossa!



Pelimuseo syntyy hartiavoimin

Suomalaisten digitaalisten pelien historia alkoi jo 1970-luvulla. Nyt on korkea aika saada ne talteen museoon – ja sinä voit auttaa siinä!

Teksti: Mikko Heinonen

Kuvat: Manu Pärssinen, Tampereen museot

Tämän vuoden maaliskuun viimeinen päivä oli minulle ja edustamalleni Pelikonepeijoonit-keräilijäryhmälle erään unelman täyttymys. Tuolloin julkistettiin yhteistyössä Tampereen kaupungin museopalveluiden ja Tampereen yliopiston kanssa synnyttämämme Suomen pelimuseo -hanke ja käynnistettiin sen joukkorahoituskampanja.

Kun avasimme Pelikonepeijoonit.net-verkkomuseomme vuonna 1999, mielessämme kajasteli lähinnä kaukainen haavekuva siitä, että jonakin päivänä sivustolla olisi myös fyysinen vastineensa. Vuosien mittaan kokoelmamme laitteet ovat olleet esillä monenlaisissa vaihtuvissa näyttelyissä, mutta pysyvä sijoituspaikka niiltä on puuttunut. Suomen pelimuseosta on vihdoin tulossa sellainen – ja paljon muutakin siinä ohessa.

Atari 2600 oli ensimmäinen supersuosittu pelikonsoli. Siihen voi tutustua aikanaan museossakin. Kuva: Reetta Tervakangas / Tampereen museot.

Vuosia kypsynyt

Ajatus pelimuseon perustamisesta ei syntynyt yhdessä yössä. Sen taustalla oli useita eri tahojen kanssa järjestettyjä näyttelyitä, runsaasti keskusteluja eri ihmisten kanssa ja yleisiä havaintoja siitä, että Suomesta puuttuu tällainen paikka. Maailmallakaan pelaamiseen keskittyviä museoita ei ole vielä ruuhkaksi asti.

Tampere valikoitui museon paikaksi useammastakin syystä. Olemme todenneet, että etenkin pelattavien laitteiden huoltaminen ja käytön opastus pitkän matkan päästä muodostuu helposti ongelmaksi. Harva haluaa muutenkaan lähettää vuosikymmenien mittaisen työnsä tulosta satojen kilometrien päähän. Siksi oli ilahduttavaa havaita, että Tampereen kaupunki oli projektissa innokkaasti mukana alusta alkaen. Kyseessä on muun muassa ensimmäinen kerta, kun julkishallinnon toimija käynnistää joukkorahoitushankkeen. Myös museolle tarjottu tila on huippuluokkaa.

Pelimuseo ei kuitenkaan ole vain Tampereen tai tamperelaisten projekti, vaan hankkeen taustavoimissa on ihmisiä kaikkialta Suomesta. Myös sijainti museokeskus Vapriikissa on erinomainen, tulipa vierailija sitten mistä päin tahansa. Rautatieasema on kävelymatkan päässä ja pysäköintitilaakin löytyy. Ja sen sijaan, että esimerkiksi lipunmyynti ja ravintola olisi tarvinnut pystyttää tämän hankkeen varoista, ovat ne nyt suoraan käytettävissä.

Lehdistön suosikki

Viimeisen kuukauden aikana pelimuseo on näkynyt paljon sekä internetissä, painetuissa lehdissä että muussakin mediassa. Itselleni suorastaan hämmentävää on ollut se, miten myönteisesti hankkeeseen on kaikkialla suhtauduttu. Esitetyt negatiivisetkin kommentit ovat pääosin olleet perusteltuja ja tuoneet esiin seikkoja, jotka olisi muutenkin ollut syytä ottaa museon suunnittelussa huomioon.

On luonnollisesti mahdotonta miellyttää kaikkia sen suhteen, mitä ensimmäiseen näyttelyyn otetaan ja mitä jätetään pois. Ohjenuorana toimii kuitenkin se, että museo kertoo suomalaisista peleistä ja pelaamisesta Suomessa. Täällä tehdyt pelit saavat paljon näkyvyyttä, mutta myös ulkomaisia ilmiöitä käsitellään niiltä osin kuin ne ovat olleet meillä merkittäviä. Etenkin pelilaitteet eli kotimikrot ja pelikonso-

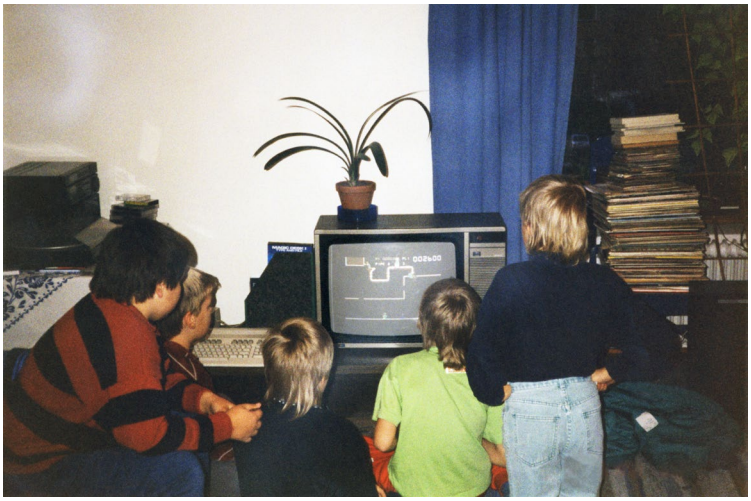
lit kun tulivat meille lähes kokonaan muualta, vaikka kotimaista ohjelmistotuotantoa olikin alusta asti.

Termi ”pelaaminen” on viime vuosina yhdistetty niin vahvasti juuri digitaaliseen pelaamiseen, että historiaa miettiessäänkin on helppo eksyä pyörittelemään ajatuksiaan vain viimeisten 45 vuoden ja erilaisten sähköisten pelien ympärillä. Tämän vuoksi työryhmässä on kiinnitetty erityistä huomiota siihen, että myös lautapelejä ja roolipelejä käsitellään kattavasti. Painopiste on kuitenkin tietoisesti digitaalisuuteen kallellaan, sillä juuri sillä alalla Suomi on kokoonsa nähden varsinainen suurvalta.

Mennyttä, nykyistä ja tulevaa

Vaikka vanhat digitaaliset pelit saavat luonnollisesti Suomen pelimuseossa paljon tilaa, museosta ei tule pelkkä retroluola. Tarkoitus on pitää sormi pelialan pulssilla ja seurata, mitä alalla tapahtuu. Tässä auttaa osaltaan myös yliopiston asiantuntemus. Pelimuseosta toivotaan kohtaamispaikkaa, jossa vanhat ja uudet pelit sekä niiden harrastajat voivat kohdata.

Tärkeintä on kuitenkin se, että museosta tulee vuorovaikutteinen. Lasin takana olevia pelejä on hieno ihailla, mutta niiden sielunelämään ei pääse tosissaan paneutumaan kokeilematta itse. Siksi haluamme rakentaa museoon useita pelipisteitä ja myös pitää ne kunnossa. Tähän vaaditaan rahaa, jota hanke kerää myymällä pääsylippuja ennakkoon.



Pelataan kuin olisi yhä vuosi 1989.
Kuva: Ilona Koivisto / Tampereen museot.

Tätä kirjoitettaessa Suomen pelimuseon joukkorahoituskampanja on kerännyt jo puolet minimitavoitteestaan. Se on hyvää vauhtia matkalla Mesenaatti.me-palvelun rahoitetuimmaksi projektiksi. Matkaa maaliin on kuitenkin vielä paljon, ja myös sinun apuasi kaivataan. Edullisimmillaan tukijaksi pääsee 10 eurolla, ja tarjolla on myös erilaisia yrityspaketteja. Tämä museo Suomesta puuttuu – tehdään se yhdessä!

Suomen pelimuseo internetissä: http://suomenpelimuseo.fi

Skrollitöllö

Teksti: Ninnu Koskenalho
Kuva: Antti Kurkinen / OUBS ry

Toukokuussa Skrollin lukijat saivat nauttia poikkeuksellisen audiovisuaalisesta skrollipläjäyksestä. Otaniemi Pop Up Living Room -niminen yhteisöllinen projekti valjasti kolmeksi toukokuun viikoksi teekkareiden OUBS-yhdistyksen televisiostudion erilaisten tapahtumien areenaksi. Skrolli oli mukana järjestämässä yleisölle avoimen keskusteluillan, jossa puhuttiin Jukka O. Kauppisen, Tapio Berschewskyn ja Ninnu Koskenalhon voimin erityisesti peleistä.

Illan vieraisiin kuuluivat 20 vuotta täyttävän Housemarque-pelitalon perustajajäsen Ilari Kuittinen, pelimusiikin veteraani Ari Pulkkinen sekä Skypen välityksellä osallistunut säveltäjä Jonne Valtonen, joka tunnetaan myös Future Crew -demoporukan Purple Motionina. Yleisöä oli mukana sekä paikan päällä että virtuaalisesti, sillä tapahtuma striimattiin livenä YouTubeen. Skrolli-yhteisö oli myös aktiivisesti mukana keskustelussa ja lähetti irc-kanavan kautta terveisiä ja kysymyksiä studioon. Kokeilimme myös livetwiittausta, ja Twitterin kautta keskusteluun saatiin terveisiä eduskunnasta asti.

Jos monimediaiset mehevät keskustelut menivät sivu suun, ei hätää: editoitu videotaltiointi on nähtävissä sivuillamme osoitteessa skrolli.fi tai keskustelufoorumilla osoitteessa www.skrolli.fi/forum/.

Ensi numerossa lisäksi luvassa laajempi reportaasi Otaniemen olohuoneen tekemisestä ja tekniikasta!

Koodaatko paremmin kuin piirrät?

Olet etsimämme henkilö.

REAKTOR.COM/CAREERS